VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Metoda Monte Carlo jako nástroj pro řešení inženýrských problémů

Monte Carlo as a tool for the solution of engineering problems

2013

Michal Béreš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání bakalářské práce

Student: **Michal Béreš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 1103R031 Výpočetní matematika

Téma: Metoda Monte Carlo jako nástroj pro řešení inženýrských problémů
Monte Carlo as a tool for the solution of engineering problems

Zásady pro vypracování:

Monte Carlo je principielně jednoduchá metoda, která slouží jako efektivní nástroj k řešení inženýrských problémů zejména tehdy, pokud analytické metody selhávají. Tato práce umožní osvojení si základních principů této metody a aplikace těchto principů k řešení konkrétních úloh.

Postup práce:
1. Studium základů simulačních metod. Princip metody Monte Carlo.
2. Simulace realizací náhodných veličin se spojitým rozdělením pravděpodobnosti.
3. Simulace realizací náhodných veličin s diskrétním rozdělením pravděpodobnosti.
4. Odhad a postižení chyb.
5. Integrace metodou Monte Carlo.
6. PC implementace metody a řešení vybraných inženýrských úloh se znázorněním a vyčíslením chyb.

Seznam doporučené odborné literatury:

- Dubi A., Monte Carlo Applications in Systems Engineering, Wiley 2000, ISBN 0-471-981729.
- R.Briš, M.Litschmannová, Statistika I, elektronické skriptum VŠB TUO, FEI,2004

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **prof. Ing. Radim Briš, CSc.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2013 .....

Rád bych na tomto místě poděkoval vedoucímu mé bakalářské práce prof. Ing. Radimu Brišovi, CSc. za čas, který mi věnoval při konzultacích, a také za zapůjčení potřebné literatury.

Abstrakt

Tato bakalářská práce se věnuje simulační metodě Monte Carlo. Hlavní části práce jsou teoretický základ metody Monte Carlo a určení chyby, přehled simulací různých spojitých a diskrétních rozdělení, integrace pomocí metody Monte Carlo včetně srovnání účinnosti s jinými numerickými přístupy a aplikace metody Monte Carlo na vybrané inženýrské úlohy, konkrétně problém průchodu neutronu materiálem a pohotovost systému v čase. Součástí práce je také implementace daných témat v jazyce Matlab.

Klíčová slova: metoda Monte Carlo, pravděpodobnost, simulace, numerická integrace

Abstract

The bachelor thesis focuses on the Monte Carlo method. The main parts are the theoretical basis of the Monte Carlo method and determination of errors, an overview of the various simulations of discrete and continuous distributions, integration using Monte Carlo method including efficiency comparsion with other numerical approaches and applications of the Monte Carlo in selected engineering problems, namely the neutron transport problem and availability modeling of systems. The work also includes the implementation of these topics in Matlab.

Keywords: Monte Carlo method, probability, simulation, numerical integration

Seznam použitých zkratek a symbolů

$A(t)$	–	pohotovost v čase t
CLV	–	Centrální limitní věta
CPU	–	Central processing unit (procesor)
CUDA	–	Compute Unified Device Architecture
$E(X)$	–	střední hodnota náhodné veličiny X
FOM	–	(Figure of Merit) účinnost dané metody
GPU	–	Graphics processing unit (grafická karta)
MMC	–	metoda Monte Carlo
MTTF	–	(mean time to failure) střední doba do poruchy
MTTR	–	(mean time to repair) střední doba do opravy
NV	–	náhodný vektor
$\mathbb{R}$	–	množina reálných čísel
U	–	makroskopický účinný průřez
$\bar{X}$	–	výběrový průměr náhodné veličiny X
ε	–	odchylka výpočtu
ξ	–	náhodné číslo, $\xi \sim U(0,1)$
σ	–	mikroskopický účinný průřez

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Při řešení složitých úloh v matematice, fyzice a jiných odvětvích se často setkáváme s neřešitelností těchto problémů klasickou analytickou cestou. V mnoha případech (vícerozměrné integrály, modelování spolehlivosti systémů, předpovídání vývoje na burze a další) se jeví jako vhodný nástroj k řešení těchto úloh právě metoda Monte Carlo. Metoda Monte Carlo je statistická stochastická metoda využívající modelování náhodných veličin.

Svůj název získala při uplatnění v simulaci života neutronů, již prováděli John von Neumann a Stanislaw Ulam, kteří použili techniku kola rulety, kdy neutron má šanci zastavit se, nebo pokračovat. K řešení složitých úloh, které byly ve své době analyticky nespočetné, byl však princip metody Monte Carlo použit již mnohem dříve. Ve Francii začátkem 18. století přírodovědec Buffon použil tuto techniku ke spočtení čísla π, avšak díky absenci počítačů bylo aplikování této metody velice časově náročné.

V dnešní době je MMC stále častěji využívána, její aplikace lze nalézt nejen v matematice či fyzice, ale také v medicíně (radiační onkologie) nebo v počítačové grafice (počítání zrcadlových odrazů).

V této práci se nejdříve seznámíme se základy teorie pravděpodobnosti a statistiky. Dále si ukážeme, jak simulovat různá diskrétní a spojitá rozdělení za použití pseudonáhodných čísel v počítači. Ukážeme princip metody Monte Carlo a odhad chyby řešení. Uvedeme princip integrace metodou Monte Carlo a zjistíme za jakých okolností je tento přístup efektivnější než jiné numerické metody. A nakonec budeme řešit aplikace metody Monte Carlo na problém průchodu neutronu materiálem a řešení pohotovosti systémů.

2 Základní poznatky z teorie pravděpodobnosti a statistiky

Tato kapitola je určena k objasnění základních principů a pojmů v pravděpodobnosti a statistice. Uvedeny jsou pouze základy nutné k pochopení další problematiky více lze nalézt v [4, 3, 5, 1].

2.1 Teorie pravděpodobnosti

2.1.1 Základní pojmy

- **náhodný pokus** – je každý děj, jehož výsledek není předem určen
- **základní prostor** – značeno Ω, je množinou všech možných výsledků náhodných pokusů
- **elementární jev** – $\{\omega\}$ je prvek základního prostoru Ω
- **náhodný jev** – A je libovolná podmnožina základního prostoru Ω

Definice 2.1 (Klasická (Laplaceova) definice pravděpodobnosti) *Je-li základní prostor konečná neprázdná množina elementárních jevů* $\{\omega_1, \ldots, \omega_n\}$, *které mají stejnou pravděpodobnost výskytu* $\left(\frac{1}{n}\right)$, *pak pravděpodobnost, že při realizaci náhodného pokusu jev A nastane, je*

$$P(A) = \frac{m}{n},$$

kde m je počet výsledků (elementárních jevů) příznivých jevu A a n počet všech možných výsledků.

- **náhodná veličina** – náhodná veličina X je reálná funkce $X : \Omega \to \mathbb{R}$ taková, že pro každé reálné x je množina
$$\{\omega \in \Omega \mid X(\omega) < x\}$$
náhodným jevem
- **pravděpodobnostní funkce** – určuje pravděpodobnost daného diskrétního jevu:
$$P(X = x_i) = P(x_i)$$
- **distribuční funkce** – nechť $F(x) : \mathbb{R} \to \mathbb{R}$ je distribuční funkcí náhodné veličiny X, pak platí:
$$F(x) = P(X < x)$$
- **hustota pravděpodobnosti** – hustota pravděpodobnosti $f(x)$ je reálná nezáporná funkce taková, že
$$F(x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t$$
pro $-\infty < x < \infty$

- **hazardní funkce** – pro nezápornou náhodnou veličinu X se spojitým rozdělením daným distribuční funkcí $F(x)$ definujeme pro $F(x) < 1$ hazardní funkci jako:

$$\lambda(x) = \frac{f(x)}{1 - F(x)}$$

- **funkce náhodné veličiny** – pro rozdělení náhodné veličiny Y, dané přepisem $Y = g(X)$, kde $g(x)$ je nějaká prostá reálná funkce definovaná na základním souboru náhodné veličiny X, platí:

$$\forall y \in \mathbb{R} : F_Y(y) = P(Y < y) = P(g(X) < y),$$

existuje-li k funkci $g(x)$ inverzní funkce $g^{-1}(x)$, pak platí:

$$F_Y(y) = \begin{cases} F_X\left(g^{-1}(y)\right) & \text{, je-li } g(x)\text{rostoucí} \\ 1 - F_X\left(g^{-1}(y)\right) & \text{, je-li } g(x)\text{klesající} \end{cases}$$

Dále si uvedeme číselné charakteristiky náhodné veličiny.

- **obecný moment r-tého řádu** – značí se $E(X^r)$ nebo μ_r, pro $r = 1, 2, 3...$

$$\mu_r = \int_{-\infty}^{\infty} x^r \cdot f(x)\,\mathrm{d}x$$

- **střední hodnota** – střední hodnota ($E(X)$, nebo μ) je parametrem náhodné veličiny, daný:

$$\mu = \int_{-\infty}^{\infty} x \cdot f(x)\,\mathrm{d}x$$

- **centrální moment r-tého řádu** – značí se $E(X - E(X))^r$ nebo μ_r', pro $r = 1, 2, 3...$

$$\mu_r = \int_{-\infty}^{\infty} (x - E(X))^r \cdot f(x)\,\mathrm{d}x$$

- **rozptyl** – rozptyl ($D(X)$, nebo σ^2) je parametrem náhodné veličiny daný:

$$\sigma^2 = \int_{-\infty}^{\infty} (x - E(X))^2 \cdot f(x)\,\mathrm{d}x,$$

při znalosti $E\left(X^2\right)$ je vhodnější použít vzorec: $D(X) = E\left(X^2\right) - (E(X))^2$

- **směrodatná odchylka** – značí se σ, spočteme ji jako: $\sigma = \sqrt{\sigma^2}$
- **kvantil** – značí se x_p, kde $p \in [0, 1]$, a platí pro něj:

$$F(x_p) = p$$

2.1.2 Základní diskrétní rozdělení

Definice 2.2 *Řekněme, že náhodná veličina X má diskrétní rozdělení pravděpodobnosti právě tehdy, když nabývá nejvýše spočetně mnoha hodnot* $\{x_1, x_2, ...\}$ *tak, že*

$$P(X = x_i) \geq 0 \; a \; \sum_{i=1}^{\infty} P(X = x_i) = 1,$$

pak funkci

$$P(X = x_i) = P(x_i)$$

nazýváme ***pravděpodobnostní funkcí*** *náhodné veličiny X a*

$$F(x) = \sum_{x_i < x} P(x_i)$$

nazýváme ***distribuční funkcí*** *náhodné veličiny X.*

Jednotlivá rozdělení a jejich vlastnosti:

- **Bernoulliho (Alternativní) rozdělení** $X \to Ber(p)$ – udává počet úspěchů v jednom pokusu, jeho pravděpodobnostní funkce, střední hodnota a rozptyl jsou:

$$P(x) = \begin{cases} p, & x = 1 \\ 1-p, & x = 0 \end{cases}; \; \mu = p; \; \sigma^2 = p(1-p)$$

- **Binomické rozdělení** $X \to Bin(n, p)$ – vzniká opakováním Bernoulliho pokusů a udává počet úspěchů v n pokusech jeho základní charakteristiky jsou:

$$P(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x}, & x \in \{0, 1, ..., n\} \\ 0, & x \notin \{0, 1, ..., n\} \end{cases}; \; \mu = np; \; \sigma^2 = np(1-p)$$

- **Geometrické rozdělení** $X \to Geom(p)$ – udává počet pokusů do prvního úspěchu, jeho základní charakteristiky jsou:

$$P(x) = (1-p)^{x-1} p, x \in \{1, 2, 3...\}; \; \mu = \frac{1}{p}; \; \sigma^2 = \frac{(1-p)}{p^2}$$

- **Hypergeometrické rozdělení** $X \to Hyp(n, r, N)$ – je definováno jako počet úspěchů v n závislých pokusech (kde r je počet příznivých možností a N je celkový počet možností), jeho pravděpodobnostní funkce je:

$$P(x) = \frac{\binom{r}{x}\binom{N-r}{n-x}}{\binom{N}{n}}, x \in \{\max\{0, r+n-N\}, ..., \min\{n, r\}\}$$

- **Negativní binomické rozdělení** $X \to NB(n,p)$ – udává počet pokusů do n-tého úspěchu, jeho základní charakteristiky jsou:

$$P(x) = \binom{x-1}{n-1} p^n (1-p)^{x-n}, x \in \{n, n+1, ...\}\,;\ \mu = \frac{n}{p};\ \sigma^2 = \frac{n(1-p)}{p^2}$$

- **Poissonovo rozdělení** $X \to Poi(\lambda t)$ – popisuje počet náhodných událostí v časovém intervalu t o četností λ, jeho základní charakteristiky jsou:

$$P(x) = \frac{(\lambda t)^x \mathrm{e}^{-\lambda t}}{x!}, x \in \{0, 1, ...\}\,;\ \mu = \sigma^2 = \lambda t$$

2.1.3 Základní spojitá rozdělení

Definice 2.3 *Řekněme, že náhodná veličina X má spojité rozdělení pravděpodobnosti právě tehdy, má-li* ***spojitou distribuční funkci***

$$F(x) = P(X < x),$$

a jelikož pravděpodobnostní funkce je nulová:

$$P(X = a) = \lim_{x \to a^+} F(x) - F(a) = F(a) - F(a) = 0,$$

zavádíme místo pravděpodobnostní funkce ***hustotu pravděpodobnosti****:*

$$f(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}.$$

Jednotlivá rozdělení a jejich vlastnosti:

- **Uniformní (rovnoměrné) rozdělení** $X \to U(a,b)$ – má konstantní hustotu pravděpodobnosti na intervalu (a,b), mimo něj nulovou. Jeho základní charakteristiky:

$$f(x) = \begin{cases} \frac{1}{b-a}, & x \in (a,b) \\ 0, & x \notin (a,b) \end{cases};\ F(x) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & x \in (a,b) \\ 1, & x \ge b \end{cases};\ \mu = \frac{a+b}{2};\ \sigma^2 = \frac{(a-b)^2}{12}$$

- **Exponenciální rozdělení** $X \to Exp(\lambda)$ – udává dobu do výskytu první události (v období stabilního života), jeho základní vlastnosti jsou:

$$f(x) = \begin{cases} \lambda \mathrm{e}^{-\lambda x}, & x > 0 \\ 0, & x \le 0 \end{cases};\ F(x) = \begin{cases} 1 - \mathrm{e}^{-\lambda x}, & x > 0 \\ 0, & x \le 0 \end{cases};\ \lambda(x) = \lambda;\ \mu = \frac{1}{\lambda};\ \sigma^2 = \frac{1}{\lambda^2}$$

- **Weibullovo rozdělení** $X \to Wei(\lambda, \beta)$ – umožňuje modelovat dobu do výskytu události (oproti exponenciálnímu rozdělení také v období časných poruch, nebo v období stárnutí):

$$f(x) = \begin{cases} \beta \lambda^\beta x^{\beta-1} \mathrm{e}^{-(\lambda x^\beta)}, & x > 0 \\ 0, & x \le 0 \end{cases};\ F(x) = \begin{cases} 1 - \mathrm{e}^{-(\lambda x^\beta)}, & x > 0 \\ 0, & x \le 0 \end{cases}$$

- **Gama rozdělení** $X \to Gama(\alpha,\lambda)$ – oproti exponenciálnímu rozdělení, které udává dobu do prvního výskytu, gama rozdělení udává dobu do α-tého výskytu (α nemusí být celé číslo), jeho hustota pravděpodobnosti je dána jako:

$$f(x) = \begin{cases} \frac{\lambda\, e^{-\lambda x}(\lambda x)^{\alpha-1}}{\Gamma(\alpha)}, & x > 0 \\ 0, & x \leq 0 \end{cases}; \alpha,\lambda > 0,$$

kde funkce Γ je rozšířením funkce faktoriál na $\mathbb{R}$, tedy pro $n \in \mathbb{N}$ platí:

$$(n-1)! = \Gamma(n) = \int_0^\infty y^{n-1} e^{-y}\, dy$$

- **Erlangova rozdělení** $X \to Erl(k,\lambda)$ – je speciálním případem gama rozdělení pro celočíselné koeficienty α, tedy Erlangovo rozdělení udává dobu do k-tého výskytu události, jeho hustota pravděpodobnosti a distribuční funkce jsou:

$$f(x) = \begin{cases} \frac{\lambda\, e^{-\lambda x}(\lambda x)^{k-1}}{(k-1)!}, & x > 0 \\ 0, & x \leq 0 \end{cases}; F(x) = \begin{cases} 1 - e^{-\lambda x}\sum_{n=0}^{k-1} \frac{(\lambda x)^n}{n!}, & x > 0 \\ 0, & x \leq 0 \end{cases}, k,\lambda > 0$$

a střední hodnota a rozptyl jsou: $\mu = \frac{k}{\lambda}; \sigma^2 = \frac{k}{\lambda^2}$

- **Normální rozdělení** $X \to N(\mu,\sigma^2)$ – je nejpoužívanějším pravděpodobnostním rozdělením modelující chování velkého množství náhodných jevů v mnoha vědních odvětvích, za určitých podmínek jím lze aproximovat řadu spojitých či diskrétních rozdělení (více v části o limitních větách), jeho specifikem je, že střední hodnota a rozptyl jsou dány jako parametry rozdělení, jeho hustota pravděpodobnosti je dána jako:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, x \in \mathbb{R},$$

jelikož hustota pravděpodobnosti normálního rozdělení není analyticky integrovatelná používá se přepočet z normovaného normálního rozdělení, jehož hodnoty jsou tabelovány:

$$X \to N(\mu,\sigma^2), Z \to N(0,1) : F_X(x) = F_Z\left(\frac{x-\mu}{\sigma}\right) = \phi\left(\frac{x-\mu}{\sigma}\right)$$

2.2 Statistika

Doposud byly uvedeny vlastnosti náhodných veličin a jejich rozdělení. V mnoha případech je však třeba určit vlastnosti náhodné veličiny (střední hodnota, rozptyl ...) z omezeného počtu jejich náhodných pokusů, tehdy se využívají metody statistické indukce. Nejdříve základní pojmy:

- **základní soubor (populace)** – množina všech prvků, které jsou sledovány
- **výběrový soubor (výběr)** – část základního souboru, kterou zkoumáme, obvykle získaná náhodným výběrem
- **náhodný výběr** – je náhodný vektor $X = (X_1,\ldots,X_n)$, který vznikne n-krát opakováním nezávislých náhodných pokusů, jeho složky jsou nezávislé náhodné veličiny X_i se stejným rozdělením pravděpodobnosti a n je rozsah náhodného výběru

2.2.1 Základní charakteristiky výběrového souboru

Nejdůležitějšími charakteristikami základního souboru jsou střední hodnota a rozptyl, zde uvedeme ekvivalentní charakteristiky výběrového souboru.

- **výběrový průměr** – je náhodná veličina definována jako:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \tag{2.1}$$

  kde $X_1, X_2 ..., X_n$ je náhodný výběr z náhodné veličiny X o rozsahu n

- **výběrový rozptyl** – je náhodná veličina dána vztahem:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2 = \frac{1}{n-1}\sum_{i=1}^{n} X_i^2 - \frac{n}{(n-1)}\bar{X^2}, \tag{2.2}$$

  kde $X_1, X_2 ..., X_n$ je náhodný výběr z náhodné veličiny X o rozsahu n, a udává míru variability výběrového souboru

2.2.2 Limitní věty

Definice 2.4 (Konvergence v pravděpodobnosti) *Řekneme, že posloupnost náhodných veličin $\{Y_n\}$ konverguje v pravděpodobnosti k náhodné veličině Y, $\left(Y_n \xrightarrow{P} Y\right)$, jestliže pro všechny $\varepsilon > 0$ pravděpodobnost, že Y_n nabude hodnoty z intervalu $(X-\varepsilon, X+\varepsilon)$ konverguje pro $n \to \infty$ k jedné , tedy:*

$$\lim_{n\to\infty} P\left[|Y_n - Y| < \varepsilon\right] = 1. \tag{2.3}$$

Definice 2.5 (Čebyševova nerovnost) *Nechť X je náhodná veličina se střední hodnotou $E(X)$ a konečným rozptylem $D(X) < \infty$, pak Čebyševova nerovnost odhaduje pravděpodobnost odchylky náhodné veličiny jako*

$$\forall \varepsilon > 0 : P\left(|X - E(X)| \geq \varepsilon\right) \leq \frac{D(X)}{\varepsilon^2}. \tag{2.4}$$

Věta 2.1 (Zákon velkých čísel) *Nechť $X_1, X_2, ..., X_n$ jsou nezávislé náhodné veličiny se stejnými středními hodnotami $E(X_1) = E(X_2) = ... = \mu$ a dále $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$, pak posloupnost $\{\bar{X}_n\}$ konverguje podle pravděpodobnosti k μ:*

$$\lim_{n\to\infty} P\left[|\bar{X}_n - \mu| < \varepsilon\right] = 1.$$

Slabý zákon velkých čísel *Nechť $X_1, X_2, ..., X_n$ jsou nekorelované náhodné veličiny se stejnými středními hodnotami $E(X_1) = E(X_2) = ... = \mu$ a konečnými rozptyly, pak:*

$$\forall \varepsilon > 0 : \lim_{n\to\infty} P\left(\frac{1}{n}\left|\sum_{i=1}^{n}\left(X_i - E(X_i)\right)\right| < \varepsilon\right) = 1.$$

Silný zákon velkých čísel *Nechť $X_1, X_2, ..., X_n$ jsou nezávislé náhodné veličiny se stejnými středními hodnotami $E(X_1) = E(X_2) = ... = \mu$, pak posloupnost $\{\bar{X}_n\}$ konverguje podle pravděpodobnosti k μ, pak:*

$$P\left(\lim_{n\to\infty} \bar{X}_n = \mu\right) = 1.$$

Věta 2.2 (Centrální limitní věta) *Nechť $X_1, X_2, ..., X_i$ jsou nezávislé náhodné veličiny se stejným (libovolným) rozdělením a s konečným rozptylem, pak výběrový průměr má při dostatečně velkém počtu pozorování přibližně normální rozdělení. Tedy:*

$$\bar{X} \sim N\left(\mu_X, \frac{\sigma_X^2}{n}\right) \text{ nebo } \frac{\bar{X} - \mu_X}{\sigma_X}\sqrt{n} \sim N(0,1),$$

kde μ_X je střední hodnota a σ_X^2 rozptyl náhodné veličiny X.

2.2.3 Další spojitá rozdělení

Nyní uvedeme další spojitá rozdělení, kterých se využívá především při odhadech a testování hypotéz ve statistice. V této práci je použijeme pro odhad chyby v MMC.

- χ^2- **rozdělení** – mějme nezávislé náhodné veličiny s normovaným normálním rozdělením $Z_1, Z_2, .., Z_n$, pak rozdělení χ_n^2 s n stupni volnosti definujeme jako:

  $$X = \sum_{i=1}^{n} Z_i^2; X \to \chi_n^2.$$

  Počet stupňů volnosti n je jediný parametrem tohoto rozdělení. Používá se především pro „test dobré shody“ a ověření nezávislosti kategoriálních proměnných.

- **Studentovo t rozdělení** – mějme dvě nezávislé náhodné veličiny: $\chi_n^2 \to \chi^2, Z \to N(0,1)$, pak Studentovo rozdělení o n stupni volnosti definujeme jako:

  $$T_n = \frac{Z}{\sqrt{\frac{\chi_n^2}{n}}}.$$

  Toto rozdělení má opět jediný parametr n, počet stupňů volnosti. Používá se především k testování hypotéz o střední hodnotě, pokud je rozptyl rozdělení neznámý.

3 Simulace spojitých a diskrétních rozdělení

V této kapitole si ukážeme principy a způsob simulace jednotlivých diskrétních a spojitých rozdělení, neboť právě vytváření náhodných pokusů je jeden z hlavních stavebních kamenů MMC. Nejprve začneme základními definicemi.

Definice 3.1 *Algoritmus nebo proces, který pro danou hustotu pravděpodobnosti $f(x)$ a distribuční funkci $F(x)$ generuje výstupní hodnoty x takové, že pro libovolné x_0 je splněna podmínka*

$$P(x < x_0) = F(x_0) \tag{3.1}$$

můžeme nazvat jako zdrojový algoritmus pro generování náhodných výběrů. [1]

Tedy platí, že náhodná proměnná realizovaná algoritmem splňujícím definici 3.1, je nerozeznatelná od náhodné proměnné realizované hustotou pravděpodobnosti dané náhodné veličiny $f(x)$, tedy

$$\int_{-\infty}^{x} f(x')\,\mathrm{d}x' = F(x) = \xi \tag{3.2}$$

řešeno pro x, kde ξ je dané náhodné číslo z uniformního rozdělení $U(0,1)$.

Věta 3.1 (Simulace distribuční funkce) *Realizace náhodné proměnné získané řešením 3.2 jsou výsledkem zdrojového algoritmu pro distribuční funkci $F(x)$. [1]*

Důkaz tohoto tvrzení je jednoduchý: nechť je x_0 náhodné reálné číslo a nechť $\xi_0 = F(x_0)$, jelikož je $F(x_0)$ neklesající funkce, platí:

$$P[x < x_0] = P[\xi < \xi_0] = \xi_0 = F(x_0).$$

Nyní máme nadefinované vlastnosti funkcí pro generování náhodných výběrů. Jelikož téměř všechen dnešní software umí generovat pseudonáhodná čísla a to z uniformního rozdělení $U(0,1)$, je nutné umět toto rozdělení převést na jiné námi požadované rozdělení. K tomuto se používá mnoho postupů, v této práci si ukážeme základní dva: inverzní transformace a metoda zamítání náhodných výběrů.

3.1 Inverzní transformace

3.1.1 Spojitá rozdělení

Jak už bylo zmíněno, pomocí generátorů pseudonáhodných čísel máme k dispozici uniformní rozdělení $U(0,1)$. Jelikož oborem hodnot všech distribučních funkcí je stejný interval, je možné této vlastnosti využít právě pro transformaci náhodné veličiny.

Definice 3.2 *Mějme náhodnou veličinu $X \in (-\infty, +\infty)$ s distribuční funkcí $F_X(x)$ a hustotou pravděpodobnosti $f_X(x)$. Jelikož $F_X(x)$ je neklesající funkce, pro libovolné $y \in (0,1)$ můžeme definovat inverzi $F_X(x)$:*

$$F_X^{-1}(y) = \inf\{x : F_X(x) > y\}. \tag{3.3}$$

V této definici bereme v potaz také možnost, že distribuční funkce $F_X(x)$ může být na nějakém intervalu $[x_s, x_d]$ konstantní, tedy hustota pravděpodobnosti $f_X(x) = 0, x \in [x_s, x_d]$, tedy

$$F_X(x) = \gamma, x \in [x_s, x_d]. \tag{3.4}$$

V tomto případě s funkcí $F_X(x)$ počítáme, jako kdyby nebyla na intervalu $(x_s, x_d]$ definována, to nás však nijak neomezuje, neboť v tomto intervalu je hustota pravděpodobnosti nulová a tedy náhodné pokusy nemohou nabývat těchto hodnot. Tedy funkce $F_X(x)$ se chová vždy jako rostoucí. Díky této vlastnosti lze dokázat, že je možné získat všechny hodnoty $X \sim F_X(x)$ pomocí hodnot $R \sim U(0,1)$. [2]
Důkaz.

Mějme číslo $R \sim U(0,1)$ a spočtěme $X = F_X^{-1}(R)$, nyní ukažme, že X je z $F_X(x)$:

$$P(X < x) = P\left(F_X^{-1}(R)\right) < x = P(R < F_X(x)) = F_X(x), \tag{3.5}$$

tedy číslo $X = F_X^{-1}(R)$ je z rozdělení $F_X(x)$.

■

Díky vztahu $X = F_X^{-1}(R)$ je tedy možné získat jakékoliv rozdělení z rozdělení $U(0,1)$. Problém ale může nastat tehdy, když není možné distribuční funkci $F_X(x)$ a její inverzi $F_X^{-1}(x)$ vyjádřit analyticky. Tento problém se dá vyřešit za použití aproximace polynomem, v tomto případě se však musí počítat s jistou nepřesností této aproximace.

3.1.2 Diskrétní rozdělení

Podobně jako v předcházejícím případě lze vycházet z vlastností distribuční funkce $F_X(x)$ a jejího oboru hodnot $[0,1]$. V případě diskrétního rozdělení je funkce $F_X(x)$ rozdělena na jednotlivé intervaly se stejnou hodnotou pravděpodobnosti, právě této vlastnosti lze využít pro simulace diskrétních rozdělení.

Mějme diskrétní rozdělení X s pravděpodobnostní funkcí definovanou jako:

$$f_k = P(X = x_k); k = 0, 1...,$$

za předpokladu seřazených hodnot náhodné proměnné X $(x_{k-1} < x_k)$ také distribuční funkci:

$$F_k = \sum_{i=0}^{k} f_i.$$

Postup získání diskrétního rozdělení se dá popsat v následujících krocích:

1. vytvoření náhodného výběru z rozdělení $R \in U(0,1)$
2. nastavení $F_0 = f_0$
3. porovnání $R \leq F$, pokud ano, pokračuje se bodem 5.
4. nastavení $F_{i+1} = F_i + f_{i+1}$ a pokračování bodem 3.
5. požadovaná hodnota $X = x_i$

Tedy procházíme distribuční funkci $F_X(x)$, dokud námi vygenerované číslo $R \in (F_{k-1}, F_k]$, pak je námi požadovaná hodnota $X = x_k$.

3.2 Metoda zamítání náhodných výběrů

V mnoha případech simulace náhodného rozdělení je použití předchozího postupu velice obtížné či časově náročné, v těchto případech je možné přistoupit k metodě založené na zamítání vygenerovaných vzorků. Hlavními přednostmi této metody jsou její jednoduchost a menší nároky na znalost distribuční funkce. Nevýhodou tohoto postupu je možná neefektivnost v případě zamítnutí velké části vygenerovaných vzorků.

Mějme hustotu pravděpodobnosti $f(x)$, jejíž rozdělení chceme simulovat na intervalu $[a,b]$, dále mějme funkci $g(x)$, jejíž rozdělení nasimulovat umíme. Nechť

$$r(x) = \frac{f(x)}{g(x)} \text{ a dále } S = \sup_{x\in[a,b]} [r(x)],$$

pak postup metody je následující:

1. Vygenerujeme x_1 z rozdělení $g(x)$.
2. Vygenerujeme náhodné číslo ξ_1, x_1 přijmeme pokud je splněna podmínka:

$$\xi_1 \leq \frac{r(x_1)}{S}.$$

   Pokud podmínka splněna není, vracíme se k bodu 1. [1]

Metoda zamítání náhodných výběrů je zdrojovým algoritmem pro generování náhodných výběrů.
Důkaz.

Rozdělení získané touto metodou se dá zapsat jako:

$$K\left(x_1 \mid \xi_1 \leq \frac{r(x_1)}{S}\right),$$

tedy dokazujeme rovnost:

$$K\left(x \mid \xi_1 \leq \frac{r(x)}{S}\right) = f(x).$$

Z definice o podmíněné pravděpodobnosti, pravděpodobnost x_1 podle $\mathrm{d}x_1$ nám dává splnění podmínky přijetí vzorku, tedy:

$$\begin{aligned} K\left(x_1 \mid \xi_1 \leq \frac{r(x_1)}{S}\right)\mathrm{d}x_1 &= \frac{g(x_1)\,\mathrm{d}x_1 \int_0^{r(x_1)/S} \mathrm{d}\xi_1}{\int_a^b \left[g(x_1)\int_0^{r(x_1)/S}\mathrm{d}\xi_1\right]\mathrm{d}x_1} = \frac{g(x_1)\,\mathrm{d}x_1\, r(x_1)/S}{\int_a^b [g(x_1)\, r(x_1)/S]\,\mathrm{d}x_1} \\ &= \frac{(f(x_1)/S)\,\mathrm{d}x_1}{\int_a^b (f(x_1)/S)\,\mathrm{d}x_1} = f(x_1)\,\mathrm{d}x_1, \end{aligned} \tag{3.6}$$

tedy je dokázáno, že metoda je zdrojovým algoritmem pro generování náhodných výběrů. [1]

■

Jak už bylo zmíněno, efektivnost této metody závisí na tom, jak dobře funkce $g(x)$ kopíruje námi požadovanou funkci $f(x)$. Efektivnost metody se dá vyjádřit jako pravděpodobnost, že bude vzorek přijat, tedy $\frac{1}{S}$.

3.2.1 Uniformní zamítání

V úvodu této části jsme ukázali princip metody zamítání, nyní si ukážeme nejednodušší realizaci této metody, a to volbou funkce $g(x)$ jako uniformního rozdělení na intervalu $[a,b]$. Mějme tedy funkci

$$g(x) = \frac{1}{b-a},$$

dále spočtěme

$$M = \sup_{x\in[a,b]} \{f(x)\} \text{ a } S = M(b-a).$$

Postup metody je tedy následující:

1. vygenerování náhodného výběru z rozdělení $g(x)$ (více v následující části) $x_1 = (b-a)\xi_1 + a$, kde ξ_1 je náhodné číslo
2. pokud platí: $\xi_2 \leq f(x_1)/M$, pak je x_1 přijato

Efektivnost této metody je zpravidla nižší než u jiných a dá se vyjádřit jako:

$$\frac{1}{S} = \frac{1}{M(b-a)}.$$

3.3 Spojitá rozdělení

V této části si ukážeme simulace jednotlivých spojitých rozdělení pomocí technik v předcházející části. Zavedeme si také pojem *náhodné číslo,* tímto označením budeme rozumět číslo $\xi \sim U(0,1)$.

3.3.1 Uniformní rozdělení

Nejjednodušší ze spojitých rozdělení, avšak pro svou jednoduchost velice často používané. Bude nás zajímat pouze interval, kde je hustota pravděpodobnosti nenulová, tedy interval (a,b). Jeho distribuční funkce:

$$F(x) = \frac{x-a}{b-a},$$

její inverze:

$$F^{-1}(\xi) = \xi(b-a) + a.$$

Náhodný pokus z uniformního rozdělení $F(x)$ získáme jako

$$x_1 = \xi_1(b-a) + a, \tag{3.7}$$

kde ξ_1 je náhodné číslo.

V Matlabu to vypadá následovně:

```
1 function [ x1 ] = U( a, b )
2 x1=(b-a)*rand(1,1)+a;
```

Výpis 3.1: Algoritmus simulace uniformního rozdělení

Toto byla nejjednodušší varianta uniformního rozdělení. Při počítání integrálů pomocí MMC se může hodit uniformní rozdělení ve vícerozměrném prostoru nebo na speciálním tvaru (kruh, koule...).

Na ukázku si uvedeme, jak simulovat *uniformní rozložení v kruhu*: budeme uvažovat polární souřadnice, tedy potřebujeme získat rozdělení pro úhel φ a poloměr r. Body v kruhu pak budou dány jako (r,φ).

Vycházíme ze vzorce pro obsah kruhu: $S=\pi r^2=\int_0^{2\pi}\int_0^R r\,\mathrm{d}r\,\mathrm{d}\varphi; \int_0^{\varphi}\int_0^r r'\,\mathrm{d}r'\,\mathrm{d}\varphi'=\frac{\varphi r^2}{2}$, pokud postupujeme podle úhlu, je postup výpočtu distribuční funkce následující:

$$\left(\frac{\varphi r^2}{2}\right)\mathrm{d}\varphi=\frac{r^2}{2}\rightarrow f_\varphi(x)=C,\int_0^{2\pi}f_\varphi(x)=1\rightarrow C=\frac{1}{2\pi}\rightarrow F_\varphi(x)=\frac{\varphi}{2\pi},$$

a jednotlivé náhodné pokusy obdržíme jako: $\varphi_1=2\pi\xi_1$. Obdobně pro obdržení distribuční funkce poloměru:

$$\left(\frac{\varphi r^2}{2}\right)\mathrm{d}r=\varphi r\rightarrow f_r(x)=Cr,\int_0^{R}f_r(x)=1\rightarrow C=\frac{2}{R^2}\rightarrow F_r(x)=\frac{r^2}{R^2},$$

tedy jednotlivé náhodné pokusy obdržíme inverzí distribuční funkce jako: $r_1=R\sqrt{\xi_1}$. Náhodné body v kruhu získáme jako $\left(R\sqrt{\xi},2\pi\xi\right)$.

Obdobným způsobem lze získat i jiné uniformní rozdělení, takovéto postupy jsou obzvlášť vhodné při počítání integrálů MMC v případě některých integračních oblastí.

3.3.2 Exponenciální rozdělení

Obdobně jako v předcházejícím případě využijeme inverzní transformaci:

$$F(x)=1-\mathrm{e}^{-\lambda t}\Rightarrow F^{-1}(\xi)=-\frac{\ln(1-\xi)}{\lambda},$$

tedy náhodný pokus z exponenciálního rozdělení můžeme vyjádřit jako:

$$x_1=-\frac{\ln(1-\xi_1)}{\lambda}, \tag{3.8}$$

kde ξ_1 je náhodné číslo. Lze také vyjádřit jako $x_1=-\ln(\xi_1)/\lambda$.

V Matlabu:

```
1 function [ x1 ] = Exp( lambda)
2 x1=-log(rand())/lambda;
```

Výpis 3.2: Algoritmus simulace exponenciálního rozdělení

3.3.3 Weibullovo rozdělení

Opět využijeme inverzní transformace:

$$F(x)=1-\mathrm{e}^{-\lambda t^\beta};F^{-1}(\xi)=\left[-\frac{\ln(1-\xi)}{\lambda}\right]^{\frac{1}{\beta}},$$

tedy náhodný pokus z Weibullova rozdělení získáme jako:

$$x_1 = \left[-\frac{\ln(\xi_1)}{\lambda} \right]^{\frac{1}{\beta}},$$

kde ξ_1 je náhodné číslo.

V Matlabu:

```
function [ x1 ] = Weib( lambda, beta )
x1=(-log(rand(1,1))/lambda)^(1/beta);
end
```

Výpis 3.3: Algoritmus simulace Weibullova rozdělení

3.3.4 Gama rozdělení

V případě gama rozdělení je simulace užitím inverzní transformace složitá a neefektivní z důvodu složitosti distribuční funkce. Je tedy nutno použít metodu zamítání náhodných výběrů. Z důvodu komplexnosti gama rozdělení je i přístup pomocí zamítání náhodných výběrů složitý a existuje více modifikací metod vzhledem k parametrům gama rozdělení, zde si uvedeme pouze gama rozdělení pro celočíselné parametry, tedy *Erlangovo rozdělení*. Více lze nalézt v [6].

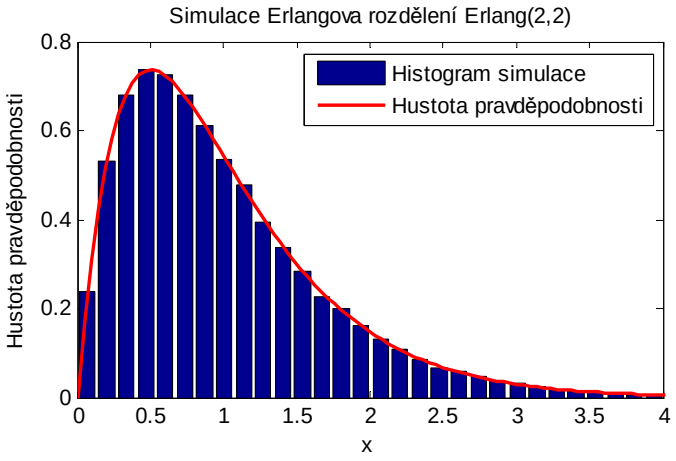
Distribuční funkce Erlangova rozdělení:

$$F(x) = 1 - \mathrm{e}^{-\lambda x} \sum_{n=0}^{k-1} \frac{(\lambda x)^n}{n!}, x \in (0, \infty).$$

Díky vlastnostem Erlangova rozdělení, konkrétně, že je sumou k exponenciálních rozdělení, lze vytvořit algoritmus na jeho simulaci pomocí simulace exponenciálního rozdělení. Tedy náhodný pokus z Erlangova rozdělení lze provést takto:

$$x_1 = \sum_{i=1}^{k} -\frac{\ln(\xi_i)}{\lambda} = -\frac{1}{\lambda}(\ln(\xi_1) + \ln(\xi)_2 + \ldots + \ln(\xi_k)) = -\frac{1}{\lambda} \ln\left(\prod_{i=1}^{k} \xi_i \right),$$

kde $\xi_1, \xi_2, \ldots, \xi_k$ jsou náhodná čísla.



Obrázek 3.1: Simulace Erlangova rozdělení s 100000 vzorky

Erlangovo rozdělení v Matlabu:

```
function [ x1 ] = Erlang( k,lambda )
x1=-1/lambda*log(prod(rand(1,k)));
```

Výpis 3.4: Algoritmus simulace Erlangova rozdělení

Ukázka simulace viz obrázek 3.1.

3.3.5 Normální rozdělení

Normální rozdělení je dalším příkladem rozdělení, kde nelze použít inverzní transformace, proto využijeme metodu zamítání náhodných výběrů.

Funkce hustoty pravděpodobnosti normálního rozdělení:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, x \in \mathbb{R}.$$

Pro jednoduchost a obecnost zvolíme jako $g(x)$ uniformní rozdělení. Jelikož pomocí této metody neumíme simulovat normální rozdělení na celé množině $\mathbb{R}$, musíme se omezit pouze na nějaký interval. Zvolený interval bude mít střed v bodě střední hodnoty normálního rozdělení a délku l tedy: $(\mu - l/2, \mu + l/2)$. Dále je třeba znát maximum $f(x)$ na daném intervalu, z vlastností normálního rozdělení je to bod, ve kterém je jeho střední hodnota tedy:

$$M = \sup_{x\in(a,b)} \{f(x)\} = f\left(\mu;\mu,\sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}},$$

potom

$$S = M(b-a) = \frac{l}{\sigma\sqrt{2\pi}}.$$

Celkový postup bude probíhat v následujících krocích:

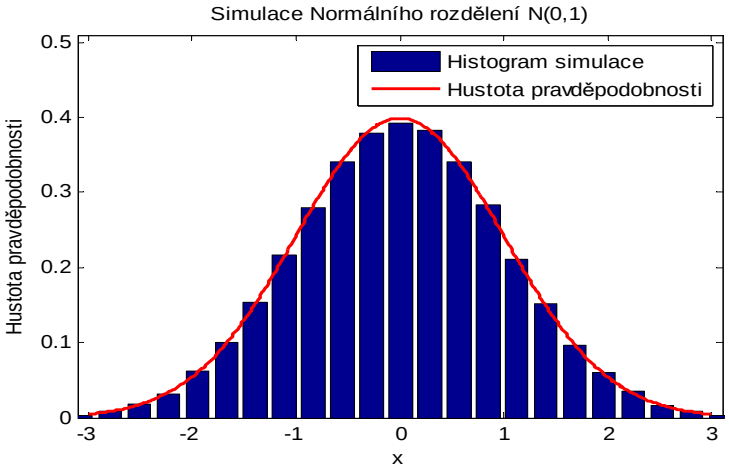
1. vytvoříme náhodný pokus $x_1 = \xi_1 l + \mu - l/2$, kde ξ_1 je náhodné číslo
2. pokud platí
$$\xi_2 \leq \mathrm{e}^{-\frac{1}{2}\left(\frac{x_1-\mu}{\sigma}\right)^2},$$
kde ξ_2 je náhodné číslo, pak je náhodný pokus přijat, jinak se vracíme k bodu 1.

Pro ukázku je uveden algoritmus přepsaný do jazyka Matlab. Značení veličin odpovídá předešlému (mu= μ, sigma= σ, rand(1,1)= ξ).

```
function [ x1 ] = N( mu, sigma,l )
x1=rand(1,1)*l + mu-l/2;
while rand(1,1)>exp(-1/2*((x1-mu)/sigma)^2)
    x1=rand(1,1)*l + mu-l/2;
end
```

Výpis 3.5: Algoritmus simulace normálního rozdělení

Ukázka simulace viz obrázek 3.2.



Obrázek 3.2: Simulace normovaného normálního rozdělení s 100000 vzorky

3.4 Diskrétní rozdělení

3.4.1 Bernoulliho rozdělení

Nejjednodušší z diskrétních rozdělení. Nabývá pouze dvou hodnot $\{0,1\}$ a jeho pravděpodobnostní funkce je dána:

$$f(x;p) = p^x(1-p)^{1-x}, x \in \{0,1\}.$$

Algoritmus simulace je následující:

1. vygenerujeme náhodné číslo ξ_1
2. pokud $\xi_1 \leq p$, pak $x_1 = 1$, jinak $x_1 = 0$

```
function [ x1 ] = Ber( p )
x1=0;
if rand(1,1)<=p
    x1=1;
end
```

Výpis 3.6: Algoritmus simulace Bernoulliho rozdělení

3.4.2 Binomické rozdělení

V případě opakování Bernoulliho pokusů dostáváme binomické rozdělení jehož pravděpodobnostní funkce je:

$$f(x;p,n) = \binom{n}{x} p^x(1-p)^{n-x}, x \in \{0,1,...,n\}.$$

Pomocí této vlastnosti lze taky binomické rozdělení simulovat:

1. vygenerujeme n náhodných pokusů z Bernoulliho rozdělení $X_1, X_2, ..., X_n \sim Ber(p)$

2. požadovaná realizace náhodného pokusu z binomického rozdělení je $x_1 = \sum_{i=1}^{n} X_i$.

```
function [ x1 ] = Bin( p, n )
X=zeros(1,n);
for i=1:n
    X(i)=Ber(p);
end
x1=sum(X);
```

Výpis 3.7: Algoritmus simulace binomického rozdělení

Tento postup je jednoduchý, avšak pro vyšší n je neefektivní. V těchto případech je možné sestrojit tabulku pravděpodobnostní funkce a následně generovat její hodnoty podle postupu v sekci Inverzní transformace pro diskrétní rozdělení. Avšak ani tento postup není vhodný pro velmi vysoké hodnoty n, pro tyto případy je vhodné použít metody založené na centrální limitní větě a aproximaci například normálním rozdělením.

Věta 3.2 (Moivreova-Laplaceova) *Nechť* $X \to Bi(n,p); \mu = np; \sigma^2 = np(1-p)$, *potom pro velká* n *platí, že:*

$$X \to N(np, np(1-p)), \text{ případně } U = \frac{X-np}{\sqrt{np(1-p)}} \to N(0,1).$$

Dle věty 3.2 pak lze použít následující algoritmus:

1. vygenerujeme náhodný pokus $Y_1 \sim N(0,1)$
2. požadovanou hodnotu získáme jako $x_1 = \max\left\{0, \left\lfloor np + \frac{1}{2} + Y_1\sqrt{np(1-p)} \right\rfloor\right\}$ [6]

3.4.3 Geometrické rozdělení

Geometrické rozdělení, definované jako počet pokusů do prvního úspěchu, je dáno pravděpodobnostní funkcí

$$f(x;p) = (1-p)^{x-1}p, x \in \{1,2,3...\}.$$

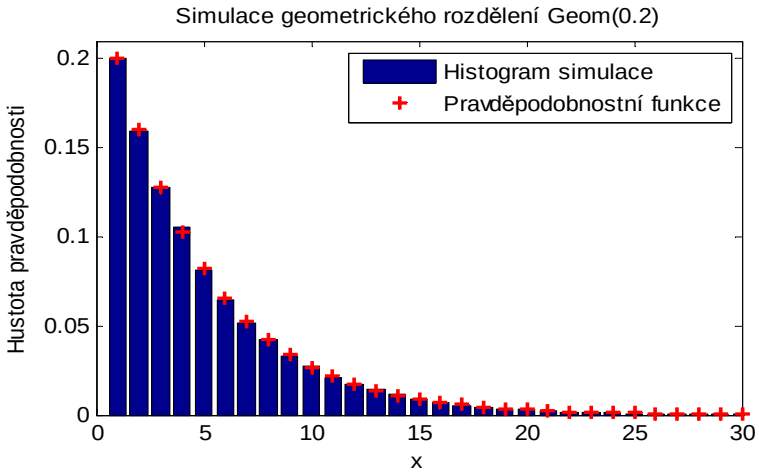
Z podobností s exponenciálním rozdělením lze získat tento vztah: nechť $Y \sim Exp(\lambda)$, kde $\lambda = -\ln(1-p)$, pak platí $\lceil Y \rceil \sim Geom(p)$. Potom využitím tohoto vztahu lze získat algoritmus pro simulaci geometrického rozdělení:

1. vygenerujeme $Y_1 \sim Exp(-\ln(1-p))$
2. náhodný pokus z geometrického rozdělení získáme jako horní celou část tohoto vzorku: $x_1 = \lceil Y_1 \rceil$

```
function [ x1 ] = Geom( p )
y1=Exp(-log(1-p));
x1=ceil(y1);
```

Výpis 3.8: Algoritmus simulace geometrického rozdělení

Ukázka simulace geometrického rozdělení viz obrázek 3.3.



Obrázek 3.3: Simulace geometrického rozdělení pro 100000 vzorků

3.4.4 Hypergeometrické rozdělení

Hypergeometrické rozdělení je definováno jako počet úspěchů v n závislých pokusech, jeho pravděpodobnostní funkce je:

$$f(x;n,r,N) = \frac{\binom{r}{x}\binom{N-r}{n-x}}{\binom{N}{n}}, x \in \langle \max\{0, r+n-N\}, \min\{n,r\} \rangle .$$

Přímo z definice hypergeometrického rozdělení lze odvodit algoritmus pro jeho simulaci:

1. vygenerujeme pole p o velikosti N, kde bude r hodnot 1 a zbytek 0
2. vygenerujeme pole náhodných čísel o velikosti N
3. toto pole setřídíme, ale zapamatujeme si původní indexy
4. vybereme prvních n hodnot a podle jejich původních indexů vybereme hodnoty z pole p
5. spočteme sumu tohoto výběru, čímž získáme x_1

Převedeno do jazyka Matlab (`poradi` je pole původních indexů) :

```
1 function [ x1 ] = Hyp( n, r, N )
2 p=zeros(1,N);
3 p(1:r)=1;
4 [~,poradi]=sort(rand(1,N));
5 x1=sum(p(poradi(1:n)));
```

Výpis 3.9: Algoritmus simulace hypergeometrického rozdělení

V případě velkých hodnot N, r je tento algoritmus neefektivní, v tomto případě využijeme vlastnosti konvergence k binomickému rozdělení:

$$Hyp(n,r,N) \to Bi\left(\frac{r}{N}, n\right).$$

3.4.5 Negativní binomické rozdělení

Negativní binomické rozdělení je definováno jako počet pokusů do n-tého úspěchu, jeho pravděpodobnostní funkce je:

$$f(x;n,p) = \binom{x-1}{n-1} p^n (1-p)^{x-n}.$$

Jelikož se dá negativně binomické rozdělení definovat jako:

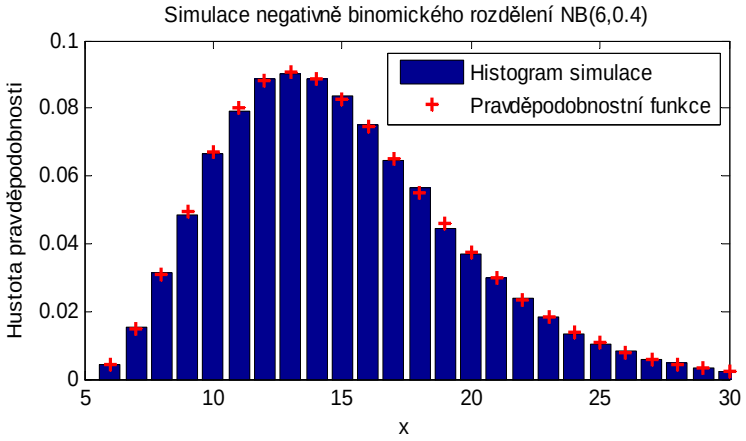
$$NB(n,p) \sim \sum_{i=1}^{n} X_i; X_1, X_2, ..., X_n \sim Geom(p),$$

pak lze simulaci $NB(n,p)$ provést takto:

1. vygenerujeme $X_1, X_2, ..., X_n \sim Geom(p)$
2. požadovaná hodnota $x_1 = \sum_{i=1}^{n} X_i$

```
1 function [ x1 ] = NB( n, p )
2 X=zeros(1,n);
3 for i=1:n
4     X(i)=Geom(p);
5 end
6 x1=sum(X);
```

Výpis 3.10: Algoritmus simulace negativně binomického rozdělení



Obrázek 3.4: Simulace negativně binomického rozdělení pro 10000 vzorků

3.4.6 Poissonovo rozdělení

Poissonovo rozdělení popisuje počet náhodných událostí v časovém intervalu t s četností λ (jejichž součin je parametrem rozdělení, proto dále budeme uvádět pouze parametr λ), jeho pravděpodobnostní funkce je následující:

$$f(x;\lambda) = \frac{(\lambda)^x e^{-\lambda}}{x!}.$$

Poissonovo rozdělení je také úzce spjato s exponenciálním rozdělením, konkrétně platí: Nechť $Y_1, Y_2, \ldots, Y_n \sim Exp(\lambda)$, pak platí:

$$X = \max\left\{ n : \sum_{i=1}^{n} Y_j \leq 1 \right\} \sim Poi(\lambda),$$

tedy Poissonovo rozdělení se dá definovat jako maximální počet náhodných pokusů exponenciálního rozdělení, jejichž suma nepřekročí 1.

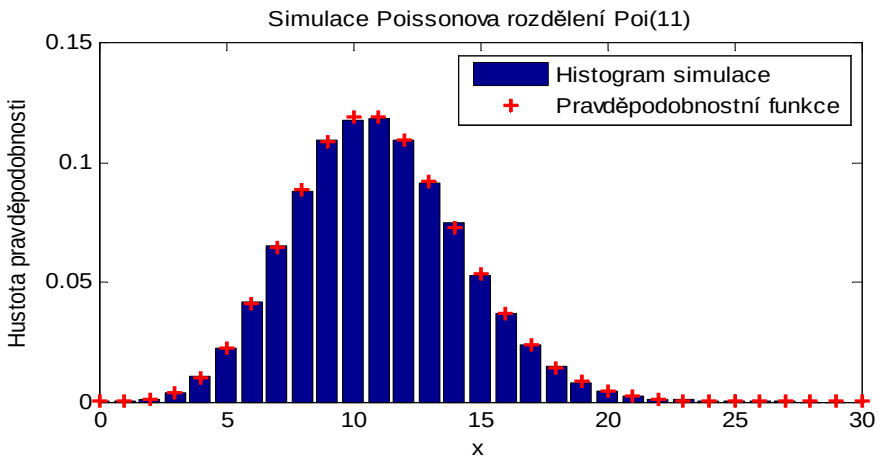
Z tohoto tvrzení lze odvodit následující postup simulace:

1. nastavíme výchozí hodnoty pomocných proměnných $n = 0$ počet pokusů a $a = 1$ hodnota součinu
2. vygenerujeme náhodné číslo ξ_n a spočteme $a = a\xi_n$
3. pokud $a \geq \mathrm{e}^{-\lambda}$ inkrementujeme $n = n + 1$ a vrátíme se ke kroku 2.
4. jinak $x_1 = n$ [6]

```
function [ x1 ] = Poi( lambda_t )
n=1;
a=1;
a=a*rand(1,1);
while a>=exp(-lambda_t)
    n=n+1;
    a=a*rand(1,1);
end
x1=n-1;
```

Výpis 3.11: Algoritmus simulace Poissonova rozdělení

Tento algoritmus se však stává neefektivním při vysokých hodnotách λ, tehdy se dá použít limitní věta nebo spočtení distribuční funkce a simulace pomocí inverzní transformace.



Obrázek 3.5: Simulace Poissonova rozdělení pro 100000 vzorků

3.4.7 Simulace diskrétních rozdělení pomocí inverzní transformace

Zatím jsme u většiny diskrétních rozdělení používali pro jejich simulace jednotlivé simulace Bernoulliho pokusů. Tento přístup je však neúčinný a časově náročný při vyšších počtech těchto pokusů pro jednu realizaci daného rozdělení. Pro vyšší hodnoty těchto pokusů, kdy však nechceme nebo nemůžeme použít aproximace spojitými rozděleními dle CLV, použijeme inverzní transformace. Tento postup spočívá ve spočtení tabulky pravděpodobnostní funkce a následném sestavení tabulky distribuční funkce, podle které se budou hodnoty simulovat.

Postup metody byl již uveden v části o inverzní transformaci, implementaci v Matlabu lze nalézt v příloze, viz výpis A.2 a implementaci funkce pro vytvoření tabulky distribuční funkce viz A.1.

Metoda pro tvorbu tabulky distribuční funkce přijímá jako parametr danou pravděpodobnostní funkci, která musí být plně definována (pro čísla mimo definiční obor vrací 0), a hranici, pro kterou má metoda ještě hodnoty zaznamenávat ε. Tento způsob byl zvolen, neboť například geometrické rozdělení má nespočetně velký definiční obor, a tedy tabulka jeho distribuční funkce by nešla sestavit.

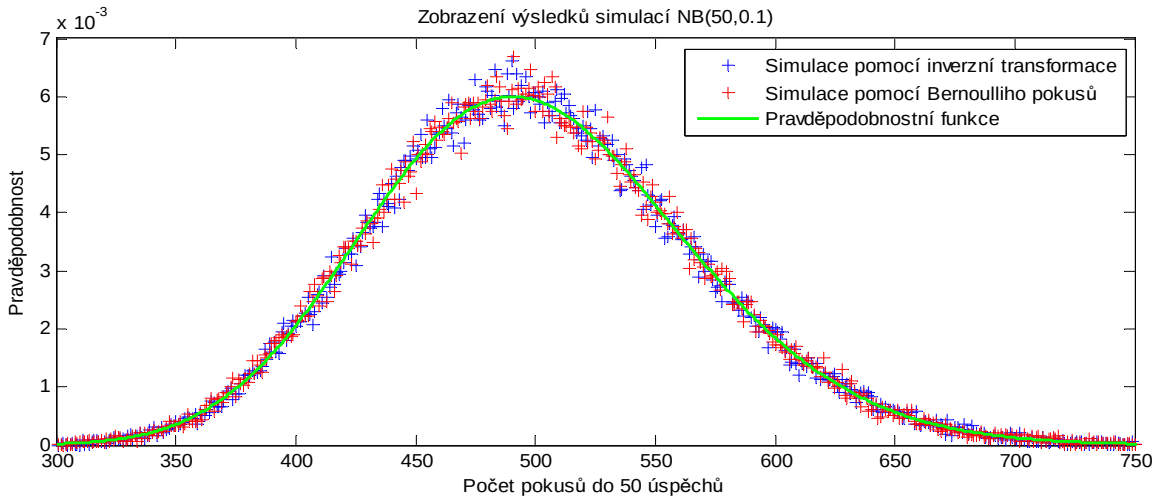
Metoda pro simulaci diskrétního rozdělení pomocí inverzní transformace přijímá jako parametry tabulku distribuční funkce a velikost požadovaného náhodného výběru. Jelikož je distribuční funkce vždy neklesající, jsou hodnoty v tabulce setříděné a díky tomu lze použít vyhledávání pomocí půlení intervalu.

Výsledky porovnávacího testu rychlosti lze vidět v následující tabulce 3.1. Z tabulky je patrné, že s rostoucím počtem pokusů je simulace pomocí Bernoulliho pokusů značně neefektivní. Oproti tomu simulace pomocí inverzní transformace je na tyto změny parametrů značně méně citlivá.

čas [s]	$n = 10, p = 0.1$	$n = 20, p = 0.1$	$n = 50, p = 0.1$
Bernoulliho pokusy	9.298071	17.308934	41.695998
Inverzní transformace	1.611819	1.718219	1.852463

Tabulka 3.1: Porovnání rychlosti simulace pomocí inverzní transformace a Bernoulliho pokusů

Na obrázku 3.6 lze vidět, že obě simulace dobře kopírují pravděpodobnostní funkci.



Obrázek 3.6: Porovnání simulace pomocí inverzní transformace a Bernoulliho pokusů

4 Odhad chyby

Předpokládejme, že řešíme pomocí MMC nějakou úlohu se skalárním výsledkem. V předchozí kapitole jsme ukázali, jak vytvářet jednotlivé náhodné výběry s daným rozdělením pravděpodobnosti, nyní si ukážeme, jak tyto hodnoty zpracovat. Hodnotu výsledku lze dostat jednoduše jako daný výběrový průměr, obtížnější je stanovit, jak vážně lze daný výsledek brát, tedy jak je přesný. Jelikož má MMC stochastický charakter, nelze vyjádřit pouze interval ve kterém se bude výsledek nacházet, ale také s jakou pravděpodobností.

V případě odhadu chyb v MMC použijeme důsledky CLV, viz věta 2.2. Konkrétně náhodná veličina definovaná jako:

$$Y = \frac{\bar{X} - \mu_X}{\sigma_X}\sqrt{n},$$

má dle CLV přibližně normální rozdělení $N(0,1)$, tedy platí pro ni následující:

$$P\left(a \leq \frac{\bar{X} - \mu_X}{\sigma_X}\sqrt{n} \leq b\right) = \phi(b) - \phi(a),$$

kde chceme, aby Y bylo v intervalu $\langle a,b\rangle$ (meze intervalu jsou kvantily normovaného normálního rozdělení) minimálně s pravděpodobností $1-\alpha$ (α je maximální přípustná pravděpodobnost, že Y je mimo daný interval), ze symetrie normálního rozdělení platí $a = -b$ tedy:

$$P\left(-b \leq \frac{\bar{X} - \mu_X}{\sigma_X}\sqrt{n} \leq b\right) \geq 1-\alpha,$$

dále si označíme maximální chybu jako:

$$\varepsilon \geq |\bar{X} - \mu_X| \Rightarrow \varepsilon = \frac{\sigma_X b}{\sqrt{n}}.$$

Nyní za použití těchto vztahů odvodíme tvar chyby:

$$\begin{aligned}
P\left(-b \leq \frac{\bar{X} - \mu_X}{\sigma_X}\sqrt{n} \leq b\right) &= \phi(b) - \phi(-b) = 2\phi(b) - 1 = 2\phi\left(\frac{\varepsilon\sqrt{n}}{\sigma_X}\right) - 1 \\
&\Downarrow \\
2\phi\left(\frac{\varepsilon\sqrt{n}}{\sigma_X}\right) - 1 &\geq 1-\alpha \\
\phi\left(\frac{\varepsilon\sqrt{n}}{\sigma_X}\right) &\geq 1 - \frac{\alpha}{2} \\
\frac{\varepsilon\sqrt{n}}{\sigma_X} &\geq z_{1-\frac{\alpha}{2}} \\
\varepsilon &\geq \frac{z_{1-\frac{\alpha}{2}}\sigma_X}{\sqrt{n}}
\end{aligned}$$

tedy nejmenší možná chyba, která splňuje předchozí nerovnosti, je dána jako:

$$\varepsilon = \frac{z_{1-\frac{\alpha}{2}}\sigma_X}{\sqrt{n}},$$

kde $z_{1-\frac{\alpha}{2}}$ je příslušný kvantil normovaného normálního rozdělení. Dále je možné vyjádřit minimální velikost náhodného výběru při dané pravděpodobnosti a chybě:

$$n = \left(\frac{z_{1-\frac{\alpha}{2}} \sigma_X}{\varepsilon} \right)^2.$$

Při využití těchto vztahů pro výpočty v MMC však narazíme na problém, a to neznámou směrodatnou odchylku σ_X, v tomto případě bychom museli použít její odhad, tedy výběrovou směrodatnou odchylku s. Zde však nelze použít aproximace normálním rozdělením, ale musíme použít rozdělení Studentovo. Avšak z vlastností Studentova rozdělení vyplývá, že při vysokém počtu stupňů volnosti se blíží normálnímu rozdělení, v praxi pro $n \geq 30$ můžeme použít místo Studentova rozdělení rozdělení normální aniž bychom se dopustili významné chyby. Díky této vlastnosti můžeme ve výpočtech pomocí MMC používat normální rozdělení, neboť velikost náhodného výběru bývá dostatečně vysoká pro aproximaci normálním rozdělením.

Hodnoty normálního normovaného rozdělení a Studentova rozdělení s daným počtem stupňů volnosti pro $99,5\%$ kvantil a jejich rozdíly viz tabulka 4.1.

$z_{99.5} = 2.5758$						
počet stupňů volnosti:	10^1	10^2	10^3	10^4	10^5	10^6
hodnota $t_{99.5,n}$	3.1693	2.6259	2.5808	2.5763	2.5759	2.5758
rozdíl hodnot v %	23.0389	1.9435	0.1912	0.0191	0.0019	0.0002

Tabulka 4.1: Porovnání normálního normovaného a Studentova rozdělení

Princip odhadu chyby si ukážeme na následujícím příkladu.

Příklad 4.1
Spočtěte, jaké přesnosti jsme dosáhli při simulaci náhodné veličiny, pokud jsme získali následující údaje: $n = 10000, \bar{X} = 1.54, s^2 = 13.69$, a požadujeme spolehlivost odhadu 95% $(\alpha = 0.05)$. Dále spočtěte, kolik náhodných pokusů ještě musíme provést, abychom dosáhli přesnosti na setiny.

Jelikož je náš náhodný výběr dostatečně velký, můžeme použít místo Studentova rozdělení, rozdělení normální. Využijeme tedy předchozího vzorce:

$$\begin{aligned} \varepsilon &= \frac{z_{1-\frac{\alpha}{2}} \sqrt{s^2}}{\sqrt{n}} \\ \varepsilon &= \frac{z_{1-\frac{0.05}{2}} \sqrt{13.69}}{\sqrt{10000}} \\ \varepsilon &= \frac{1.96 \times 3.7}{100} \\ \varepsilon &\doteq 0.0725 \end{aligned}$$

Spočetli jsme odhad chyby, tedy námi spočtená střední hodnota simulované náhodné veličiny leží v intervalu $(1.54 - 0.0725, 1.54 + 0.0725)$ s 95% pravděpodobností.

Dále spočteme velikost náhodného výběru nutnou pro maximální odchylku $\varepsilon = 0.01$.

$$\begin{aligned} n &= \left(\frac{z_{1-\frac{\alpha}{2}}\sigma_X}{\varepsilon}\right)^2 \\ n &= \left(\frac{z_{1-\frac{0.05}{2}}\sqrt{13.69}}{0.01}\right)^2 \\ n &= \left(\frac{1.96 \times 3.7}{0.01}\right)^2 \\ n &= (725.2)^2 \\ n &= 525915.04 \end{aligned}$$

Tedy pro požadovanou přesnost bychom museli provést ještě 515916 pokusů.

△

Z příkladu vidíme, že pro rostoucí přesnost potřebujeme velké množství náhodných pokusů, což bývá časově náročné. Přesněji, například pokud chceme interval spolehlivosti zmenšit na polovinu, musíme vytvořit čtyřikrát více náhodných pokusů.

5 Princip metody Monte Carlo

Jak už bylo zmíněno, MMC využívá modelování náhodných veličin, tedy k řešenému příkladu vytvoříme náhodnou veličinu, jejíž střední hodnota je shodná s řešením původního problému. Lze tedy řešit nejen stochastické, ale také deterministické problémy, např. výpočet integrálu. Ačkoliv má náhodná veličina shodné řešení s daným problémem, nejsme schopni toto řešení získat, neboť můžeme provést pouze spočetně mnoho náhodných pokusů. Z těchto náhodných pokusů sestrojíme výběrový průměr, který je pouze aproximací střední hodnoty.

Mějme úlohu, jejímž přesným výsledkem je hodnota θ, obecný postup řešení úloh pomocí MMC je tedy následující:

- sestrojíme náhodnou veličinu X, jejíž střední hodnota je shodná s výsledkem zadané úlohy θ
- vytvoříme požadovaný počet náhodných pokusů $X_1, X_2, ..., X_n$ náhodné veličiny X, toto se provádí simulací náhodné veličiny X pomocí náhodných čísel v počítači, viz sekce 3.
- vyhodnocení získaného náhodného výběru, tedy určení výběrového průměru $\bar{X}$ a vyhodnocení chyby (stanovení spolehlivosti odhadu a spočtení daného spolehlivostního intervalu) viz sekce 4.

Tento postup si ukážeme na následujících příkladech.

Příklad 5.1
Spočtěte číslo π.

Řešení: Toto je jedna z nejčastějších ukázkových úloh pro MMC, která má také dlouhou historii. Už v 18. století počítal Georges Louis Leclerc de Buffon číslo π stochastickou metodou, tehdy však pro výpočet požíval vlastnosti pravděpodobnosti dopadu jehly na papír s vodorovnými čárami. Tato metoda je fyzicky jednodušeji proveditelná, my však použijeme počítač a spokojíme se s jednoduchým geometrickým odvozením.

V obrázku 5.1 vidíme čtvrtinu jednotkového kruhu uvnitř jednotkového čtverce, popsáním jejich obsahů se dá vyjádřit vztah pro výpočet π, tedy

$$S_{čtverec} = a^2, S_{kruh} = \frac{\pi \cdot r^2}{4},$$

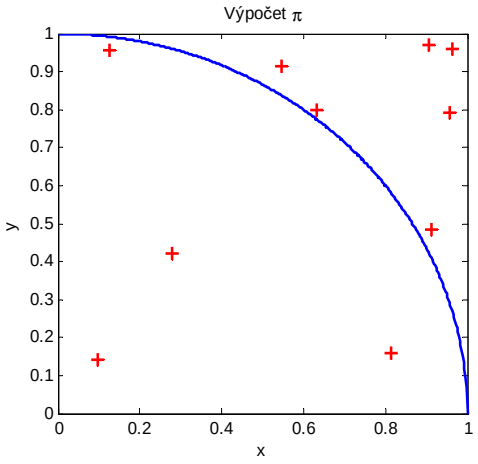
pro $a = r = 1$:

$$\frac{S_{kruh}}{S_{čtverec}} = \frac{\frac{\pi \cdot r^2}{4}}{a^2} = \frac{\pi}{4} \Rightarrow \pi = \frac{4 \cdot S_{kruh}}{S_{čtverec}}. \tag{5.1}$$

Nyní vytvoříme náhodnou veličinu X definovanou:

$$x_i = \begin{cases} 1 & -\text{uvnitř kruhu} \\ 0 & -\text{mimo kruh} \end{cases}; \ \mu_X = \frac{\pi}{4},$$

tato veličina obsahuje všechny body v daném jednotkovém čtverci a náhodným pokusům přiřazuje podle polohy hodnoty 1 nebo 0. Zde jsme nevytvořili náhodnou veličinu, která přesně odpovídá řešení dané úlohy, toto však není velký problém neboť střední hodnota naší náhodné veličiny lze lehce převést na hodnotu požadovanou. Formulace náhodné veličiny je tedy hotova.



Obrázek 5.1: Znázornění principu výpočtu π

Jako další fázi je třeba generovat náhodné pokusy této veličiny. Je zjevné, že se jedná o spojitou náhodnou veličinu s uniformním rozdělením:

$$f(x) = \begin{cases} 1, & x \in (0,1) \times (0,1) \\ 0, & x \notin (0,1) \times (0,1) \end{cases},$$

což nasimulujeme jednoduše jako body, jejichž souřadnice jsou náhodná čísla. Zdrojový kód této simulace v Matlabu viz výpis 5.1.

```
function [ NV ] = pi_simulace( n )
NV=zeros(1,n);                      % nahodny vektor
for i=1:n
    point=rand(2,1);                % generace bodu
    if ((point(1)^2+point(2)^2)<1) % zjisteni polohy bodu
        NV(i)=1;
    else
        NV(i)=0;
    end
end
```

Výpis 5.1: Generování náhodných pokusů pro spočtení čísla π

Pomocí tohoto kódu můžeme vytvořit náhodný výběr libovolné velikosti, tedy další část je hotova.

V poslední části musíme zpracovat získaný náhodný výběr, k čemuž nám poslouží následující funkce v Matlabu, která pro vstupní náhodný výběr a požadovanou míru spolehlivosti spočte výběrový průměr a odchylku.

```
function  vysledek= vyhodnoceni(NV, alpha)
% NV - pole s nahodnym vyberem
% alpha - maximamalni pripustna pravdepodobnost chyby
n=length(NV);                       % pocet nahodnych pokusu
t_alpha=tinv(1-alpha/2,n);          % prislusny kvantil stud. r.
E_x=0;                              % obec. moment 1. radu
```

```
E_x2=0;                                % obec. moment 2. radu
for i=1:n
    E_x=E_x+NV(i);
    E_x2=E_x2+NV(i)^2;
end
E_x=E_x/n;
E_x2=E_x2/n;                           % vyberovy prumer
s2=n/(n-1)*(E_x2-E_x^2);               % vyberovy rozptyl
epsilon=t_alpha*sqrt(s2)/sqrt(n);      % chyba
vysledek=[ E_x epsilon s2 n E_x2];
```

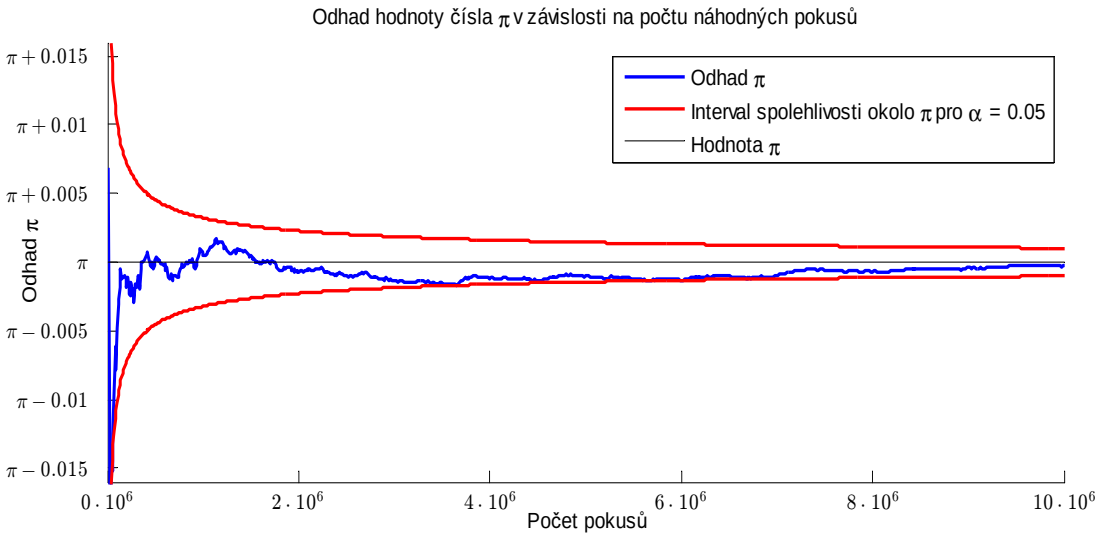
Výpis 5.2: Funkce pro vyhodnocení náhodného výběru

Pro univerzálnost této funkce používáme kvantily Studentova rozdělení, což však při vyšších hodnotách není třeba. Dále si uvedeme tabulku spočtených hodnot s jejími spočtenými odchylkami a skutečnými chybami odhadu.

velikost náhodného výběru	10^3	10^4	10^5	10^6	10^7
výběrový průměr	3.1040	3.1248	3.1420	3.1393	3.1413
spočtená odchylka	0.1035	0.0324	0.0102	0.0032	0.0010
absolutní rozdíl od čísla π	0.0376	0.0168	0.0004	0.0022	0.0003

Tabulka 5.1: Spočtené odhady čísla π pro α=5%

V tabulce vidíme, že spočtená odchylka byla vždy menší než reálná chyba odhadu. Také lze vidět, že odhad se zpřesňuje s odmocninou počtu náhodných pokusů. Pro lepší ukázku si ještě uvedeme graf (obrázek 5.2) se znázorněnými odhady pro dané velikosti náhodných vektorů a jejich příslušným intervalem spolehlivosti kolem přesné hodnoty.



Obrázek 5.2: Zobrazení odhadů čísla π a jejich intervalů spolehlivosti pro $\alpha = 0.05$

△

Dále si ukážeme příklad, který se k názvu Monte Carlo více hodí, bude se jednat o hru v kostky.

Příklad 5.2
Jedna z oblíbených her s kostkami se nazývá Vrhcáby. Princip této hry je poměrně jednoduchý: hází se šesti kostkami v kolech po třech hodech, hráč si vybírá z předem stanovených kombinací (postupka, samé šestky, atd...) a po každém hodu si může dát stranou kostky, které se mu hodí, a zbylými házet znovu, pokud nepoužije všechny tři hody, může si zbytek uložit pro příště. Jak lze vidět, nejedná se pouze o štěstí, ale i dobrý úsudek. Hráč musí vědět, kolik přibližně potřebuje hodů na danou kombinaci, a případně se rozhodnout, o kterou se bude snažit, podle toho, co už naházel.

V této úloze tedy budeme počítat střední hodnoty počtu hodů různých kombinací.

Samé šestky (Generál) Nyní určíme střední hodnotu počtu hodů pro naházení šesti šestek od prvního hodu. Náhodná veličina X tedy udává počet hodů kostkami do padnutí samých šestek (hází se jen těmi, na kterých šestka není) a nabývá hodnot $x \in \{1,2,3...\}$, její náhodný výběr získáme postupnou simulací hodů kostkami. Postup simulace této náhodné veličiny lze vidět v algoritmu viz výpis 5.3.

Tento algoritmus je obecný, funguje pro různé požadované kombinace a různé počáteční stavy i v případě házení více než šesti kostkami. Využívá funkci `dice`(n), která vytvoří simulaci hodu n kostkami.

```
function [ NV ] = vrhcaby( p, h, p_k, n )
% p - pole s pocty pozadovanych hodnot pr. [0 0 0 0 0 6]
% h - pole jiz hozenych kostek pr. [1 0 2 3 0 0]
% p_k - pocet kostek kterymi hazeme
% n - pozadovana velikost nahodneho vyberu
NV=zeros(1,n);                  % nahodny vyber
for k=1:6                       % odlozeni vyhovujicich kostek
    if h(k)>p(k)                % az v poli h zustanou pouze pocty
        h(k)=p(k);              % odlozenych kostek
    end
end
for i=1:n
    zbyva=p_k-sum(h);           % pocet kostek kterymi hazeme
    h_t=h;                      % pole hodnot poctu odlozenychkostek
    hod=0;                      % pocet hozenych hodu
    while (sum(p)-sum(h_t))>0   % dokud nehodime danou komb
        hozeno=dice(zbyva);     % simulace hodu zbylymi k.
        for j=hozeno            % pro kazdou hozenou hodnotu
            if h_t(j)<p(j)      % kontrola zdali ji chceme
                h_t(j)=h_t(j)+1;
            end
        end
        hod=hod+1;
        zbyva=p_k-sum(h_t);
    end
    NV(i)=hod;
end
```

Výpis 5.3: Simulace počtu hodů prodanou kombinaci

Pomocí náhodného výběru o velikosti $n = 100000$ a při intervalu spolehlivosti $\alpha = 0.05$ jsme získali tyto hodnoty: $\bar{X} = 13.9334, \varepsilon = 0.0415$. Tedy střední hodnota počtu hodů do naházení všech šestek se nachází v intervalu $(13.8919, 13.9749)$.

Nakonec uvedeme tabulku středních hodnot počtu hodů různých kombinací za předpokladu, že házíme od prvního hodu.

	Generál $\{6,6,6,6,6,6\}$	Postupka $\{1,2,3,4,5,6\}$	Malý dům $\{1,2,3,4,5\}$	Velký dům $\{2,3,4,5,6\}$	Pyramida $\{4,5,5,6,6,6\}$
$\bar{X}$	13.9334	7.9282	4.5975	4.6003	10.1270
ε	0.0415	0.0350	0.0180	0.0181	0.0383

Tabulka 5.2: Střední hodnoty počtu hodů pro dosažení daných kombinací $(\alpha = 0.05)$

△

6 Integrace pomocí metody Monte Carlo

V této části si ukážeme možnosti uplatnění MMC při řešení integrálů. Základní přístupy k řešení integrálů pomocí MMC jsou dva:

- **geometrická interpretace** – vychází z interpretace integrálu jako oblasti pod integrovanou funkcí (1D - plocha, 2D - objem, ...), tento princip nedosahuje vysoké efektivnosti, využitím tohoto principu jsme počítali číslo π, viz příklad 5.1
- **interpretace integrálu jako náhodné veličiny** – využívá přepis integrálu na náhodnou veličinu, jejíž střední hodnotu zjišťujeme, touto metodou se budeme nadále zabývat

6.1 Principy a řešení jednorozměrných integrálů

Je dán integrál:

$$I = \int_a^b g(x)\,\mathrm{d}x,$$

kde $g(x)$ je libovolná funkce integrovatelná na $[a,b]$. Dále mějme hustotu pravděpodobnosti $f(x)$, pro kterou platí, pokud $g(x) \neq 0$, pak $f(x) > 0$. Potom můžeme daný integrál zapsat jako:

$$I = \int_a^b \left[\frac{g(x)}{f(x)}\right] f(x)\,\mathrm{d}x = \int_a^b h(x)\,f(x)\,\mathrm{d}x, \tag{6.1}$$

dále zadefinujme náhodnou veličinu X s rozdělením pravděpodobnosti $f(x)$ a náhodnou veličinu $Y = h(X)$. Pak vzorec 6.1 je definicí střední hodnoty náhodné veličiny Y:

$$I = E(Y).$$

Tedy integrál je vyjádřen jako střední hodnota náhodné veličiny Y, dále využijeme vlastností výběrového průměru a získáme:

$$I \approx \tilde{I}_n = \bar{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i,$$

tedy odhad integrálu získáme jako výběrový průměr náhodného výběru z Y. Chybu odhadu lze vyjádřit jako:

$$\varepsilon = \frac{t_{1-\frac{\alpha}{2},n}\sqrt{s_n^2}}{\sqrt{n}},$$

kde $t_{1-\frac{\alpha}{2},n}$ je kvantil Studentova rozdělení o n stupních volnosti a α je hladina významnosti (maximální pravděpodobnost větší chyby než ε).

Jelikož efektivita metody záleží na vhodné volbě hustoty pravděpodobnosti $f(x)$, zavedeme parametry pro porovnávání těchto efektivností. Pro vyjádření relativní přesnosti (Relativní Standardní Odchylka - RSO) použijeme vzorec pro odhad chyby:

$$RSO = \frac{\sqrt{s_n^2}}{\bar{X}\sqrt{n}}.$$

Pro porovnávání efektivnosti dvou metod nás však zajímá také čas, za který jsme schopni odhad provést tedy:

$$T = n \cdot \tau,$$

kde τ je čas pro provedení jednoho náhodného pokusu. Zkombinováním těchto vztahů získáme vztah pro celkovou efektivnost metody (angicky *Figure of Merit - FOM*):

$$FOM = \frac{1}{RSO^2 \times T} = \frac{\bar{X}^2}{s_n^2 \tau}.$$

V případě porovnávání více metod na jedné úloze je $\bar{X}$ přibližně stejný, tedy efektivnost metody dokážeme zvýšit snížením rozptylu nebo času potřebného pro výpočet.

Vyjádříme-li si rozptyl náhodné veličiny Y:

$$\begin{aligned}
\sigma_Y^2 &= E\left(Y^2\right) - \left(E\left(Y\right)\right)^2 = \int_a^b h\left(x\right)^2 f\left(x\right) \mathrm{d}x - I^2 = \int_a^b \left[\frac{g\left(x\right)}{f\left(x\right)}\right]^2 f\left(x\right) \mathrm{d}x - I^2 \\
&= \int_a^b \left[\frac{g\left(x\right)}{f\left(x\right)}\right]^2 f\left(x\right) \mathrm{d}x - I^2 \int_a^b f\left(x\right) \mathrm{d}x = \int_a^b \left[\left(\frac{g\left(x\right)}{f\left(x\right)}\right)^2 - I^2\right] f\left(x\right) \mathrm{d}x.
\end{aligned}$$

Jelikož efektivita roste se snižujícím se rozptylem, je pro nulový rozptyl maximální. Tedy hustota pravděpodobnosti $f\left(x\right)$ bude ideální, bude-li splňovat následující vztah:

$$\begin{aligned}
\left(\frac{g\left(x\right)}{f\left(x\right)}\right)^2 - I^2 &= 0 \\
\left(\frac{g\left(x\right)}{f\left(x\right)}\right)^2 &= I^2 \\
\frac{g\left(x\right)}{f\left(x\right)} &= I \\
f\left(x\right) &= \frac{g\left(x\right)}{I}.
\end{aligned} \tag{6.2}$$

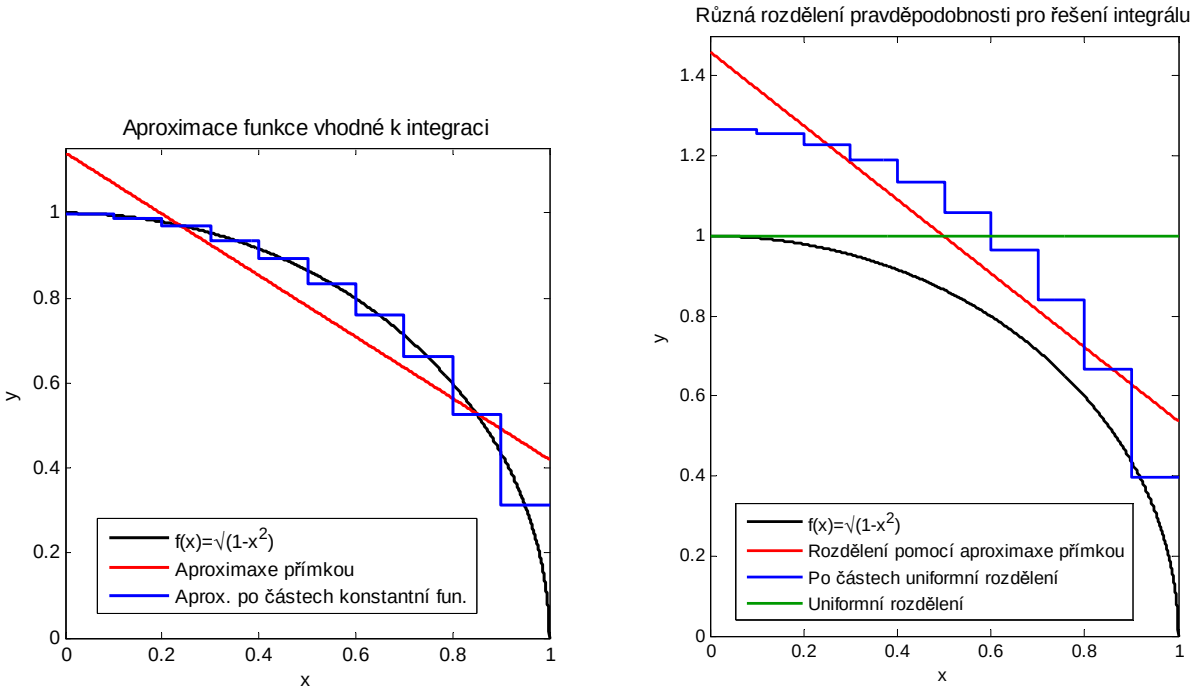
Jak je vidět, ideální hustota pravděpodobnosti ztrácí smysl, neboť pro její konstrukci musíme znát hodnotu integrálu. Ale z rovnice 6.2 lze také usoudit, že efektivnost metody bude růst, pokud hustota pravděpodobnosti bude kopírovat funkci $g\left(x\right)$. Toto je však komplexní problém, který se dá řešit mnoha způsoby. Některé tyto způsoby a porovnání jejich účinnosti si ukážeme v následujícím příkladě.

Příklad 6.1
Spočtěte následující integrál:

$$I = \int_0^1 \sqrt{1 - x^2} \, \mathrm{d}x.$$

Jak je vidět, tento příklad je ekvivalentní s příkladem 5.1, kde jsme počítali číslo π pomocí geometrické interpretace, na závěr tohoto příkladu přidáme její výsledky k porovnání s jinými metodami. Jak už jsme

si ukázali, při integraci pomocí MMC záleží na vhodné volbě pravděpodobnostního rozdělení. Zde si vyzkoušíme řešení zadaného integrálu pomocí třech druhů rozdělení: uniformního, po částech uniformního a rozdělení s hustotou pravděpodobnosti ve tvaru polynomu. Grafické znázornění pravděpodobnostních rozdělení použitých k integraci lze vidět na obrázku 6.1b.



(a) Aproximace integrované funkce

(b) Možná pravděpodobnostní rozdělení vhodná pro integraci

Obrázek 6.1: Aproximace a pravděpodobností rozdělení pro integraci

Uniformní rozdělení: Toto je nejjednodušší způsob řešení integrálů pomocí náhodné veličiny. Všechny hodnoty z daného integračního intervalu jsou vybírány se stejnou pravděpodobností. Velkou výhodou je, že nemusíme řešit vlastnosti funkce a generování náhodných výběrů z tohoto rozdělení je velmi rychlé, nevýhodou může být menší efektivita ve formě většího výběrového rozptylu. Pro jednorozměrný integrál lze výpočet zapsat jako:

$$I \approx \tilde{I}_n = \bar{X}_n = \frac{(b-a)}{n} \sum_{i=1}^{n} f(x_i); x_i \sim U(a,b),$$

kde a, b jsou integrační meze a n je velikost náhodného výběru.

Po částech uniformní rozdělení: Anglicky *„piecewise distribution“*. Jedná se v podstatě o vymezení částí, ve kterých chceme různě hustě generovat body. Výhodou je značná redukce výběrového rozptylu oproti uniformnímu rozdělení (v závislosti na počtu intervalů), nevýhodou je však vyšší náročnost při generování bodů a vyhodnocování funkce. Další značnou nevýhodou je geometricky rostoucí počet

rozdělení při rostoucí dimenzi integrálu, stejně jako při numerické integraci (více si uvedeme v další části).

Pro výpočet integrálu pomocí tohoto rozdělení potřebujeme vytvořit funkci pro generování náhodných výběrů z daného rozdělení a také funkci reprezentující hustotu pravděpodobnosti. Jelikož vyjádření hustoty pravděpodobnosti tohoto rozdělení je již prováděno při simulaci tohoto rozdělení, je vhodné spojit generování bodů s vyjádřením hodnoty funkce hustoty pravděpodobnosti. Tento postup lze vidět ve výpisu C.3.

Rozdělení s hustotou pravděpodobnosti ve tvaru polynomu: V tomto případě zvolíme polynom 1. stupně, jak uvidíme, použití vyšších stupňů by bylo velmi složité. Nejdříve musíme zvolit vhodný polynom. Tento problém se dá řešit například použitím Taylorova polynomu, nebo aproximací integrované funkce polynomem. V této úloze využijeme aproximaci polynomem. Pro řešení aproximace byl navržen algoritmus využívající metody nejmenších čtverců, viz výpis B.1, tento algoritmus je schopen řešit i aproximace funkcí více proměnných. Tímto se však budeme zabývat v dalších částech této práce.

Předpokládejme tedy, že máme polynom $p(x) = cx + d$, který je na intervalu (a,b) kladný, a jeho integrál na tomto intervalu je roven 1 jako na obrázku 6.1b. Pak můžeme příslušnou hustotu pravděpodobnosti založenou na tomto polynomu zapsat jako:

$$f(x) = \begin{cases} cx+d, & x \in (a,b) \\ 0, & x \notin (a,b) \end{cases},$$

Dále spočítat hustotu pravděpodobnosti jako:

$$F(x) = \begin{cases} 0, & x \in (-\infty, a] \\ \int_a^x ct + d\,\mathrm{d}t, & x \in (a,b) \\ 1, & x \in [b,\infty) \end{cases} = \begin{cases} 0, & x \in (-\infty, a] \\ \frac{cx^2}{2} + dx - \left(\frac{ca^2}{2} + da\right) & x \in (a,b) \\ 1, & x \in [b,\infty) \end{cases}.$$

Pro generování náhodného výběru z tohoto rozdělení je třeba vyjádřit inverzi distribuční funkce $F(x)$. Jelikož nás zajímá pouze interval, na kterém je hustota pravděpodobnosti nenulová, pro spočtení inverze řešíme následující rovnici:

$$F(x) = \xi; \xi = \frac{cx^2}{2} + dx - \left(\frac{ca^2}{2} + da\right), \tag{6.3}$$

kde $x \in (a,b)$. Pro určení inverze vyjádříme z rovnice 6.3 x:

$$\frac{c}{2}x^2 + dx - \left(\frac{ca^2}{2} + da + \xi\right) = 0$$

z tohoto tvaru získáme řešení:

$$x = \frac{-d \pm \sqrt{d^2 - 2c\left(\frac{ca^2}{2} + da + \xi\right)}}{c}, \tag{6.4}$$

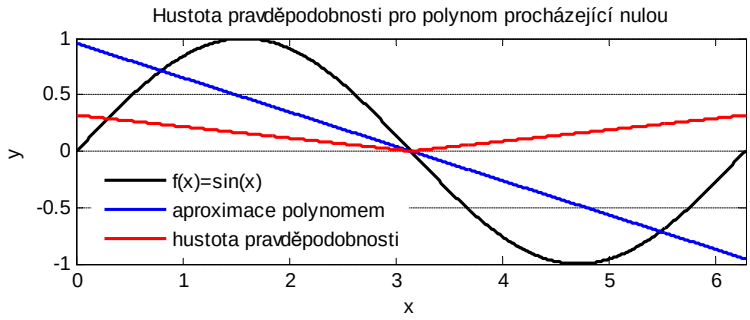
vztah 6.4 má více než jedno řešení, proto jej upravíme na:

$$cx + d = \pm\sqrt{d^2 - 2c\left(\frac{ca^2}{2} + da + \xi\right)}.$$

Zde již vidíme, že z požadavků kladnosti pravděpodobnostní funkce musí být řešení, a tedy i inverze distribuční funkce:

$$F^{-1}(\xi) = x = \frac{-d + \sqrt{d^2 - 2c\left(\frac{ca^2}{2} + da + \xi\right)}}{c}.$$

Je třeba dodat, že toto je neefektivní postup, použitý pouze pro ukázku. Při aproximaci můžeme dostat polynomy, které na intervalu (a,b) procházejí nulou, pak by se náhodný výběr z tohoto rozdělení generoval složitěji, viz obrázek 6.2.



Obrázek 6.2: Ukázka hustoty pravděpodobnosti založené na polynomu procházejícím nulou

Výsledky jednotlivých metod při řešení daného integrálu a jejich spočítané účinnosti testované pro náhodné výběry o velikosti 100000 lze vidět v tabulce 6.1. Je vidět, že nejvhodnější hustota pravděpodobnosti pro daný integrál je po částech uniformní, a to s co největším počtem dělení, což také vyplývá z vyšší efektivnosti obdélníkové metody oproti MMC v integraci pro 1D. Ve vyšších dimenzích by tato hustota pravděpodobnosti byla méně účinná než uniformní z důvodu náročného generování náhodných výběrů. Je také vidět, že ačkoliv je geometrická interpretace velmi rychlá, je její účinnost horší než u uniformní hustoty pravděpodobnosti. To společně s nároky na geometrickou interpretaci (nutná znalost maxima funkce) činí tento způsob nevhodným pro integraci.

	$\bar{X}$	ε	$t\,[s]$	FOM
Uniformní r.	0.7858	0.0014	1.6165	$7.7316 \cdot 10^5$
Po částech uniformní r. (10 dělení)	0.7853	0.00036	3.1355	$59.671 \cdot 10^5$
Po částech uniformní r. (100 dělení)	0.7854	0.00007	4.4805	$1018 \cdot 10^5$
Polynomiální r.	0.7852	0.0006	3.7168	$16.445 \cdot 10^5$
Geometrická interpretace	0.7841	0.0026	0.5097	$7.1255 \cdot 10^5$

Tabulka 6.1: Test integrace pomocí MMC s různými hustotami pravděpodobnosti

△

6.1.1 Řešení nevlastních integrálů

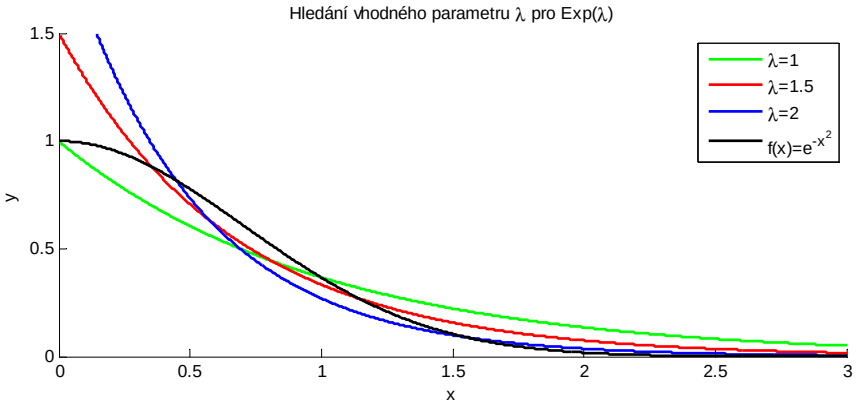
Pomocí MMC se dají také řešit nevlastní integrály. Obyčejný numerický přístup při řešení nevlastních integrálů je najít mez, za kterou je již hodnota integrálu zanedbatelná. MMC je díky svým vlastnostem schopná řešit problém jinak, stačí nám pravděpodobnostní rozdělení, které má nenulovou hustotu

pravděpodobnosti na intervalu stejném, jako jsou dané integrační meze. Pro nevlastní integrály se nabízí exponenciální rozdělení, které má tento interval $(0,\infty)$. Ukázka řešení integrálu za použití tohoto rozdělení viz příklad 6.2.

Příklad 6.2
Spočtěte následující integrál:

$$I = \int_0^\infty \mathrm{e}^{-x^2}\,\mathrm{d}x.$$

Všimněme si, že integrační meze odpovídají intervalu, na kterém má exponenciální rozdělení nenulovou hustotu pravděpodobnosti. Volíme tedy exponenciální rozdělení. Otázkou však zůstává, jaký zvolíme jeho parametr, aby integrace MMC byla co možná nejefektivnější. Tento problém lze řešit například minimalizací rozdílu hodnot integrované funkce a hledané hustoty pravděpodobnosti, avšak tento příklad slouží pouze jako ukázka integrace nevlastních integrálů, proto se spokojíme s odhadem parametru podle grafu funkcí. Na obrázku 6.3 vidíme tři křivky reprezentující exponenciální rozdělení s různými parametry. Nejlépe kopíruje integrovanou funkci hustota pravděpodobnosti exponenciálního rozdělení s parametrem $\lambda = 1.5$.



Obrázek 6.3: Exponenciální rozdělení s rozdílnými parametry pro vhodnou integraci

Pro výpočet daného integrálu budeme postupovat klasicky z definice integrace na začátku kapitoly. Využijeme přitom funkci pro generování hodnot z exponenciálního rozdělení z kapitoly 3.

Při výpočtu pro náhodný výběr o velikosti $n = 100000$ s intervalem spolehlivosti $\alpha = 0.05$ jsme získali následující hodnoty: $\bar{X} = 0.8864$ a $\varepsilon = 0.0016$. Což se blíží skutečné hodnotě, která je $\frac{\sqrt{\pi}}{2} \doteq 0.886227$.

△

6.2 Řešení vícerozměrných integrálů

Už bylo zmíněno, že integrace pomocí MMC se hodí zvláště pro vícerozměrné integrály. Toto vyplývá z vlastností chyby v MMC, která klesá v závislosti na $(\sqrt{n})^{-1}$ bez ohledu na dimenzi integrálu. Naproti tomu klasické numerické metody, jako jsou například Gaussovy kvadratury, jsou silně závislé na dimenzi

integrálu. Konkrétně pro numerickou integraci platí, že chyba integrace klesá v závislosti $n^{-m/k}$, kde m je závislé na daném integračním pravidlu (v případě kvadratur $m \sim 3$) a k je dimenze integrálu. Tedy pro dimenze 6 a vyšší je MMC vhodnějším nástrojem než jiné numerické metody. [2]

Pro integraci ve více dimenzích platí totéž, co pro jednoduchou integraci. Je dán integrál:

$$I = \int_{\Omega} g(\vec{x})\,\mathrm{d}\Omega,$$

kde $\Omega \subset \mathbb{R}^n, \vec{x} \in \Omega$. Mějme dále hustotu pravděpodobnosti $f(\vec{x})$, pro kterou platí, že pokud $g(\vec{x}) > 0$, pak $f(\vec{x}) \neq 0$ pro $\vec{x} \in \Omega$. Pak platí následující vztah:

$$I = \int_{\Omega} \left[\frac{g(\vec{x})}{f(\vec{x})}\right] f(\vec{x})\,\mathrm{d}\Omega = \int_{\Omega} h(\vec{x})\, f(\vec{x})\,\mathrm{d}\Omega.$$

Dále mějme náhodnou veličinu X s hustotou pravděpodobnosti $f(\vec{x})$ a náhodnou veličinu Y definovanou jako $Y = h(X)$. Pak střední hodnota náhodné veličiny je rovna hodnotě integrálu a integrál můžeme odhadnout pomoci vyběrového průměru náhodné veličiny Y:

$$I \approx \tilde{I}_n = \bar{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

Jak je vidět, vícerozměrná integrace je stejná jako jednorozměrná. Opět vhodná volba hustoty pravděpodobnosti ovlivní náš výpočet. V případě vícerozměrných integrálu je však používání jiného než uniformního rozdělení obtížné a mnohdy nevýhodné z důvodu dlouhého výpočetního času. Proto se budeme věnovat pouze integraci pomocí uniformního rozdělení.

Uniformní rozdělení ve vícerozměrném prostoru Je v podstatě složeno z n uniformních rozdělení, kde n je rozměr prostoru. Generování hodnot z tohoto rozdělení se provádí postupným generováním hodnot jednotlivých rozměrů. Tedy pokud $\Omega = (a,b) \times (c,d) \times (e,f)$, pak $\vec{x} = (x_1, x_2, x_3)$, kde $x_1 \sim U(a,b), x_2 \sim U(c,d), x_3 \sim U(e,f)$. Algoritmus generování náhodného výběru z uniformního rozdělení v n-rozměrném prostoru lze vidět ve výpise 6.1.

```
function [ NV ] = U_mult( o, n )
% funkce pro generovani nahodneho vyberu z uniformniho r. v n-dim. prostoru
% o - pole urcujici oblast omega pr. omega=(a,b)x(c,d)->o=[a b c d]
% n - pozadovana velikost nah. vyberu
k=length(o)/2;       % urceni dim prostoru
NV=rand(k,n);        % nahodne body v prostoru (0,1)
for i=1:k            % transformace do daneho prostoru pro jednu slozku NV
    NV(i,:)= NV(i,:)*(o(2*i)-o(2*i-1))+o(2*i-1);
end
```

Výpis 6.1: Simulace uniformního rozdělení ve vícerozměrném prostoru

Samotné počítání integrálu pak probíhá klasicky podle předchozích vzorců, viz výpis C.1. Dále si ověříme efektivnost metody v závislosti na rozměru integrálu a porovnáme s Gaussovými kvadraturami. Pro tento účel byl vytvořen algoritmus pro vícerozměrnou integraci pomocí Gaussových kvadratur, viz výpis C.2.

Příklad 6.3
Otestujte MMC a Gaussovy kvadratury na následujících sadách integrálů:

$$\int_{-2}^{2} \mathrm{e}^{-x^2}\,\mathrm{d}x, \int_{-2}^{2}\int_{-2}^{2} \mathrm{e}^{-x^2-y^2}\,\mathrm{d}x\,\mathrm{d}y, \int_{-2}^{2}\int_{-2}^{2}\int_{-2}^{2} \mathrm{e}^{-x^2-y^2-z^2}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \ldots$$

$$\int_{0}^{3} \sqrt{x}\,\mathrm{e}^{-x}\,\mathrm{d}x, \int_{0}^{3}\int_{0}^{3} \sqrt{xy}\,\mathrm{e}^{-x-y}\,\mathrm{d}x\,\mathrm{d}y, \int_{0}^{3}\int_{0}^{3}\int_{0}^{3} \sqrt{xyz}\,\mathrm{e}^{-x-y-z}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \ldots$$

Pro řešení těchto integrálů použijeme již uvedené algoritmy: C.1 a C.2. Budeme řešit dané integrály pro dimenze 1...10 a zaznamenávat výsledky jednotlivých metod. Při použití MMC použijeme náhodné výběry o velikosti $n = 100000$ a odhad spočteme s hladinou významnosti $\alpha = 0.05$. Při řešení za použití Gaussových kvadratur volíme kvadratury 5. stupně a chybu počítáme jako odchylku od správného řešení (tento přístup může pro jisté integrály Gaussovy kvadratury zvýhodnit, počítání maximální přípustné chyby by však bylo velice složité). Přesné hodnoty integrálů se dají určit díky symetričnosti integrálů skrze jednotlivé dimenze. Tedy v našem případě platí:

$$\left(\int_{-2}^{2} \mathrm{e}^{-x^2}\,\mathrm{d}x\right)^2 = \int_{-2}^{2}\int_{-2}^{2} \mathrm{e}^{-x^2-y^2}\,\mathrm{d}x\,\mathrm{d}y,$$

a obdobně pro další dimenze. Tento fakt vyplývá ze shodných integračních mezí, stejné formulace pro všechny dimenze a faktu, že se formule jedné proměnné chová jako konstanta při integraci dle jiné proměnné. Avšak ani základní integrály nejsou integrovatelné, avšak jejich hodnoty jsou známé:

$$\int_{-2}^{2} \mathrm{e}^{-x^2}\,\mathrm{d}x = \sqrt{\pi}Erf\,(2); \int_{0}^{3} \sqrt{x}\,\mathrm{e}^{-x}\,\mathrm{d}x = \frac{\sqrt{\pi}}{2}Erf\left(\sqrt{3}\right) - \frac{\sqrt{3}}{\mathrm{e}^3},$$

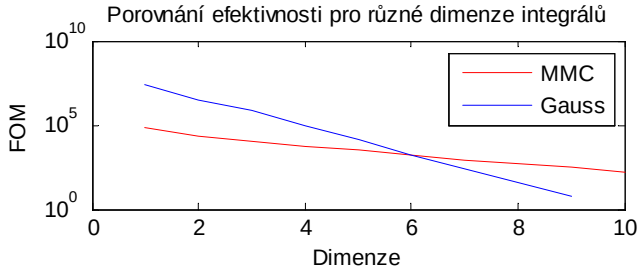
kde $Erf\,(x)$ je Gaussova chybová funkce. První integrál je Gaussova funkce a druhý je gama funkce.

Nyní již přistoupíme k testům.

	MMC			Gauss		
dimenze	$\bar{X}$	ε	$t\,[s]$	$\tilde{I}$	ε	$t\,[s]$
1	1.7731	0.0085	0.6614	1.7736	0.0094	0.0014
2	3.1101	0.0244	0.7082	3.1455	0.0333	0.0032
3	5.4877	0.0607	0.7620	5.5788	0.0883	0.0059
4	9.7541	0.1451	0.7928	9.8945	0.2082	0.0236
5	17.1229	0.3324	0.8598	17.5485	0.4604	0.1140
6	30.2787	0.7646	0.8931	31.1235	0.9773	0.5541
7	52.1659	1.6670	1.1029	55.1997	2.0169	2.8318
8	97.2638	4.1190	1.1459	97.9004	4.0774	14.1659
9	162.5271	8.7924	1.1462	173.6331	8.1141	81.7147
10	299.5123	21.0271	1.2285	-	-	-

Tabulka 6.2: Srovnání výpočtu MMC a Gaussovými kvadraturami pro první sadu integrálů

Z tabulky lze vidět, že odchylky od přesné hodnoty jsou pro obě funkce přibližně stejné, avšak čas pro výpočet pomocí Gaussových kvadratur exponenciálně roste. Toto je způsobeno exponenciálním růstem počtu spočtených hodnot na síti pro integrační oblast v Gaussových kvadraturách. Dále si ještě uvedeme graf efektivity obou metod, na kterém je dobře vidět rozměr integrálu, pro který se stává MMC účinější.

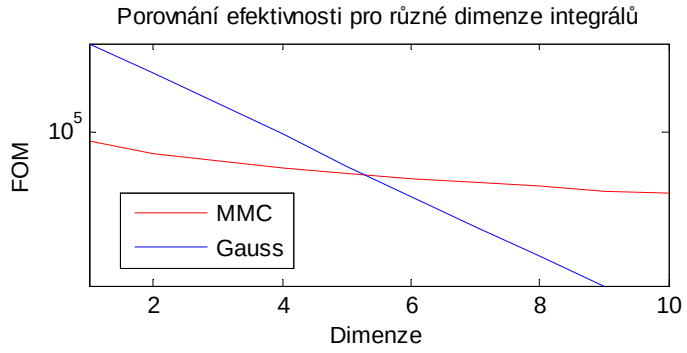


Obrázek 6.4: Porovnání efektivity MMC a Gaussových kvadratur pro první sadu integrálů

Dále si uvedeme tabulku a graf výsledků pro druhou sadu integrálů.

	MMC			Gauss		
dimenze	$\bar{X}$	ε	$t\,[s]$	$\tilde{I}$	ε	$t\,[s]$
1	0.7877	0.0021	2.4389	0.7909	0.0036	0.0018
2	0.6188	0.0025	2.4024	0.6255	0.0057	0.0027
3	0.4891	0.0025	2.4650	0.4947	0.0067	0.0083
4	0.3868	0.0024	2.5226	0.3913	0.0070	0.0353
5	0.3029	0.0022	2.5732	0.3095	0.0069	0.1749
6	0.2389	0.0020	2.6327	0.2447	0.0066	0.8855
7	0.1877	0.0018	2.6894	0.1936	0.0061	4.4433
8	0.1480	0.0016	2.7730	0.1531	0.0055	22.6906
9	0.1151	0.0014	3.2553	0.1211	0.0048	113.2012
10	0.0918	0.0012	2.8583	-	-	-

Tabulka 6.3: Srovnání výpočtu MMC a Gaussovými kvadraturami pro druhou sadu integrálů



Obrázek 6.5: Porovnání efektivity MMC a Gaussových kvadratur pro druhou sadu integrálů

Jak je vidět ze spočtených a změřených hodnot, účinnost Gaussových kvadratur exponenciálně klesá s rostoucím rozměrem integrálu. MMC je opravdu lepší volbou pro integrály 6 a vyšší dimenze, jak už jsme uvedli na začátku této části.

△

6.2.1 Vícerozměrná integrace přes netriviální oblasti

Při řešení vícerozměrných integrálů často narazíme na integrační oblasti, které nejsou zadané jako oblast ve tvaru $((a,b)\times(b,d)\times ...)$, ale například koule, elipsoid a jiné. I v těchto případech lze použít MMC.

Jednou možností je vytvoření vlastního uniformního rozdělení pokrývající danou oblast. Toto je obecně obtížný úkol, ale v případě jednoduše definovaných tvarů (koule...) je efektivní. Princip si ukážeme na příkladě.

Příklad 6.4
Spočtěte integrál:

$$\iiint_{\Omega} \mathrm{e}^{-x^2-y^2-z^2}\,\mathrm{d}\Omega; \Omega = \left\{x,y,z\in\mathbb{R} : x^2+y^2+z^2 \leq 1\right\}.$$

Jak je vidět, jedná se o integrál přes objem jednotkové koule. Úloha lze samozřejmě řešit také analyticky, zde si ve zkratce uvedeme řešení:

$$\iiint_{\Omega} \mathrm{e}^{-x^2-y^2-z^2}\,\mathrm{d}\Omega = \int_0^1\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\int_0^{2\pi} \cos(\vartheta)\, r^2\, \mathrm{e}^{-r^2}\,\mathrm{d}\varphi\,\mathrm{d}\vartheta\,\mathrm{d}r = \tag{6.5}$$

$$4\pi\int_0^1 r^2\,\mathrm{e}^{-r^2}\,\mathrm{d}r = 2\pi\int_0^1 \sqrt{t}\,\mathrm{e}^{-t}\,\mathrm{d}t = \sqrt{\pi^3}Erf(1) - \frac{2\pi}{\mathrm{e}} \sim 2.38098. \tag{6.6}$$

Při výpočtu jsme nejprve provedli transformaci do sférických souřadnic (vzorec 6.5) a dále substituci (vzorec 6.6), kterou vznikla gama funkce.

Dále si ukážeme řešení MMC, nejprve odvodíme generování hodnot z uniformního rozdělení přes kouli:

$$\int_0^r\int_{-\frac{\pi}{2}}^{\vartheta}\int_0^{\varphi} \cos\left(\vartheta'\right) r'^2\,\mathrm{d}\varphi'\,\mathrm{d}\vartheta'\,\mathrm{d}r' = \frac{r^3}{3}\varphi\left(\sin(\vartheta)+1\right); \left(r=R,\varphi=2\pi,\vartheta=\frac{pi}{2}\right) \to \frac{4}{3}\pi R^3 \tag{6.7}$$

ve vzorci 6.7 jsme vyjádřili funkci pro spočtení objemu koule jako funkci horní meze integrálu. Tento vzorec využijeme pro odvození generování jednotlivých rozměrů.

$$\left(\frac{r^3}{3}\varphi\left(\sin(\vartheta)+1\right)\right)\mathrm{d}r = r^2\varphi\left(\sin(\vartheta)+1\right) \to f_r(r) = C_r r^2, \int_0^R C_r r^2\,\mathrm{d}r = 1 = \left[\frac{C_r r^3}{3}\right]_0^R$$

$$\Rightarrow C_r = \frac{3}{R^3} \quad ; \quad F_r(r) = \frac{r^3}{R^3} \to r = R\sqrt[3]{\xi} \tag{6.8}$$

$$\left(\frac{r^3}{3}\varphi\left(\sin\left(\vartheta\right)+1\right)\right)\mathrm{d}\vartheta \quad = \quad \frac{r^3}{3}\varphi\left(\cos\left(\vartheta\right)\right)\rightarrow f_\vartheta(\vartheta)=C_\vartheta\cos\left(\vartheta\right),\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}C_\vartheta\cos\left(\vartheta\right)\mathrm{d}\vartheta=1$$

$$\left[C_\vartheta\sin\left(\vartheta\right)\right]_{-\frac{\pi}{2}}^{\frac{\pi}{2}}=1 \quad \Rightarrow \quad C_\vartheta=\frac{1}{2};F_\vartheta\left(\vartheta\right)=\frac{1}{2}\sin\left(\vartheta\right)+\frac{1}{2}\rightarrow\vartheta=\arcsin\left(2\xi-1\right) \tag{6.9}$$

$$\left(\frac{r^3}{3}\varphi\left(\sin\left(\vartheta\right)+1\right)\right)\mathrm{d}\varphi \quad = \quad \frac{r^3}{3}\left(\sin\left(\vartheta\right)+1\right)\rightarrow f_\varphi(\varphi)=C_\varphi,\int_0^{2\pi}C_\varphi\,\mathrm{d}\varphi=1=\left[C_\varphi\varphi\right]_0^{2\pi}$$

$$\Rightarrow C_\varphi=\frac{1}{2\pi} \quad ; \quad F_\varphi\left(\varphi\right)=\frac{\varphi}{2\pi}\rightarrow\varphi=2\pi\xi \tag{6.10}$$

Ze vzorců (6.8, 6.9, 6.10) můžeme vyjádřit bod z uniformního rozdělení v kouli ve sférických souřadnicích jako

$$y=\left(R\sqrt[3]{\xi_1},\arcsin\left(2\xi_1-1\right),2\pi\xi_1\right),$$

kde ξ_1,ξ_2,ξ_3 jsou náhodná čísla. Převod do Kartézských souřadnic se provede jako:

$$x=\left(y_1\sin\left(y_2\right)\cos\left(y_3\right),y_1\sin\left(y_2\right)\sin\left(y_3\right),y_1\cos\left(y_2\right)\right).$$

Nyní dokážeme simulovat uniformní rozdělení pro oblast ve tvaru koule, pro řešení integrálu musíme ještě znát hustotu pravděpodobnosti daného rozdělení. Ta se v našem případě spočítá jednoduše jako:

$$f\left(\vec{x}\right)=\begin{cases}\frac{1}{V_{koule}}=\frac{3}{4\pi r^3}, & \vec{x}\in\Omega\\ 0, & \vec{x}\notin\Omega\end{cases}.$$

Na základě předchozích poznatků lze sestavit algoritmus pro danou integraci, který můžeme vidět ve výpisu C.4. Nyní vyřešíme zadaný integrál, zvolíme velikost náhodného výběru $n=100000$ a hladinu významnosti $\alpha=0.05$. Hodnoty získané výpočtem jsou $\bar{X}=2.3821$ a $\varepsilon=0.004$, tedy hledaná hodnota integrálu bude s 95% pravděpodobností v intervalu $(2.3781,2.3861)$. Což odpovídá analyticky spočtené hodnotě.

△

Další možnost, jak generovat uniformní rozdělení na dané oblasti Ω, je pomocí metody zamítání náhodných výběrů. Pro tento postup nám stačí vícerozměrné uniformní rozdělení definované na oblasti $\Omega_1\supset\Omega$, kde $\Omega_1=(a,b)\times(c,d)\times...$ a funkce která jednoznačně určí, zdali $\vec{x}\in\Omega$.

Avšak chceme-li integrovat přes danou oblast, musíme znát hustotu pravděpodobnosti použitého rozdělení. V tomto případě je hledaná hodnota na dané oblasti konstantní a je rovna:

$$f\left(\vec{x}\right)=\frac{1}{V_\Omega},\vec{x}\in\Omega.$$

Zde však může nastat problém, neboť musíme znát objem oblasti Ω, což je obecně obtížné spočítat. Nabízí se využít hodnot získaných pomocí metody zamítání náhodných výběrů, konkrétně počet přijatých výběrů n_p a celkový počet výběrů n_c, a spočítat odhad V_Ω. Tedy:

$$\tilde{V_\Omega}=\bar{X_n}=\frac{1}{n}\sum_{i=1}^{n}X_i=V_{\Omega_1}\frac{n_p}{n_c};X=\begin{cases}V_{\Omega_1}, & \vec{x}\in\Omega\\ 0, & \vec{x}\notin\Omega\end{cases}.$$

Z tohoto vztahu získáme odhad pro objem dané oblasti. Nabízí se možnost, jak toto spojit do jediného výpočtu (vyhneme se tak složitějšímu odhadu výsledné chyby z odhadu objemu a výsledných hodnot). Pro tento způsob přeformulujeme původní integrál:

$$\int_{\Omega} g(\vec{x})\,\mathrm{d}\Omega = \int_{\Omega_1} g'(\vec{x})\,\mathrm{d}\Omega_1, \Omega \subset \Omega_1, g'(\vec{x}) = \begin{cases} g(\vec{x}) & \vec{x} \in \Omega \\ 0 & \vec{x} \notin \Omega \end{cases},$$

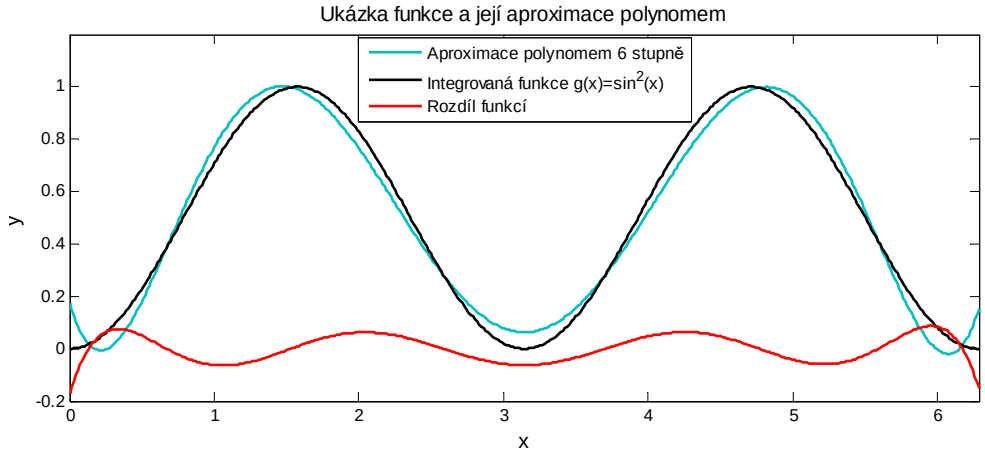
zvolíme-li $\Omega_1 = (a,b) \times (c,d) \times \ldots$, pak pro generování hodnot můžeme zvolit vícerozměrné uniformní rozdělení jako v části o vícerozměrných integrálech. Algoritmizovanou verzi tohoto postupu můžeme vidět ve výpisu C.5. Lze vidět, že se jedná pouze o jednoduchou úpravu původního kódu pro vícerozměrnou integraci (C.1).

6.3 Vymezení hlavní části

Na začátku této části jsme si ukázali zvýšení efektivity integrace pomocí MMC volbou vhodné hustoty pravděpodobnosti. Toto však není jediný způsob, jak zpřesnit výpočet integrálu pomocí MMC. Další často užívanou metodou je vymezení hlavní části. Tato metoda je principiálně jednoduchá: máme-li integrovat funkci a známe přitom jinou funkci kopírující průběh funkce integrované, jejíž integrál na daném intervalu je známý, pak stačí MMC integrovat pouze rozdíl těchto funkcí. Tento způsob zapíšeme jako:

$$I = \int_a^b g(x)\,\mathrm{d}x, I_1 = \int_a^b f(x)\,\mathrm{d}x : I = \int_a^b g(x) - f(x)\,\mathrm{d}x + I_1,$$

za předpokladu, že funkce $f(x)$ dostatečně kopíruje integrovanou funkci $g(x)$ je rozptyl při výpočtu MMC menší a tedy i výsledný interval spolehlivosti kratší. Pro ilustraci na obrázku 6.6 vidíme funkci $g(x) = \sin(x)$ a její aproximaci polynomem. Je také vidět, že rozptyl při integraci rozdílů těchto funkcí bude znatelně menší než při původní formulaci integrálu.



Obrázek 6.6: Aproximace integrované funkce pro vymezení hlavní části

Jako vhodný způsob pro použití metody vymezení hlavní části se jeví aproximace polynomem, či Taylorův rozvoj, protože polynom lze jednoduše mechanicky integrovat díky jednoduchým integračním

pravidlům. Algoritmus této integrace můžeme vidět ve výpisu B.2. S využitím aproximace polynomem pomocí funkce B.1 si na následujícím příkladě ukážeme rostoucí přesnost dvourozměrné integrace MMC s vymezením hlavní části.

Příklad 6.5
Pro integrál:

$$I = \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \sin^2(x)\cdot\sin^2(y)\,\mathrm{d}x\,\mathrm{d}y$$

použijte MMC a vymezení hlavní části pro zpřesnění výpočtu.

K tomuto příkladu použijeme již zmíněnou funkci pro aproximaci polynomem. Použijeme aproximace polynomy $2,4,6,8$ a 10 stupně, každý spočtený pro síť o straně 100 a náhodný výběr velikosti $n = 100000$.

Výsledky lze vidět v následující tabulce:

	bez	2 stupně	4 stupně	6 stupně	8 stupně	10 stupně
ε	0.0686	0.0667	0.0464	0.0307	0.0219	0.0110
čas $[s]$	0.7142	0.8358	1.4576	2.6718	5.1873	6.7834
FOM	$1.12\cdot10^5$	$1.01\cdot10^5$	$1.2\cdot10^5$	$1.49\cdot10^5$	$1.51\cdot10^5$	$4.54\cdot10^5$

Tabulka 6.4: Integrace MMC s vymezením hlavní části pomocí polynomu

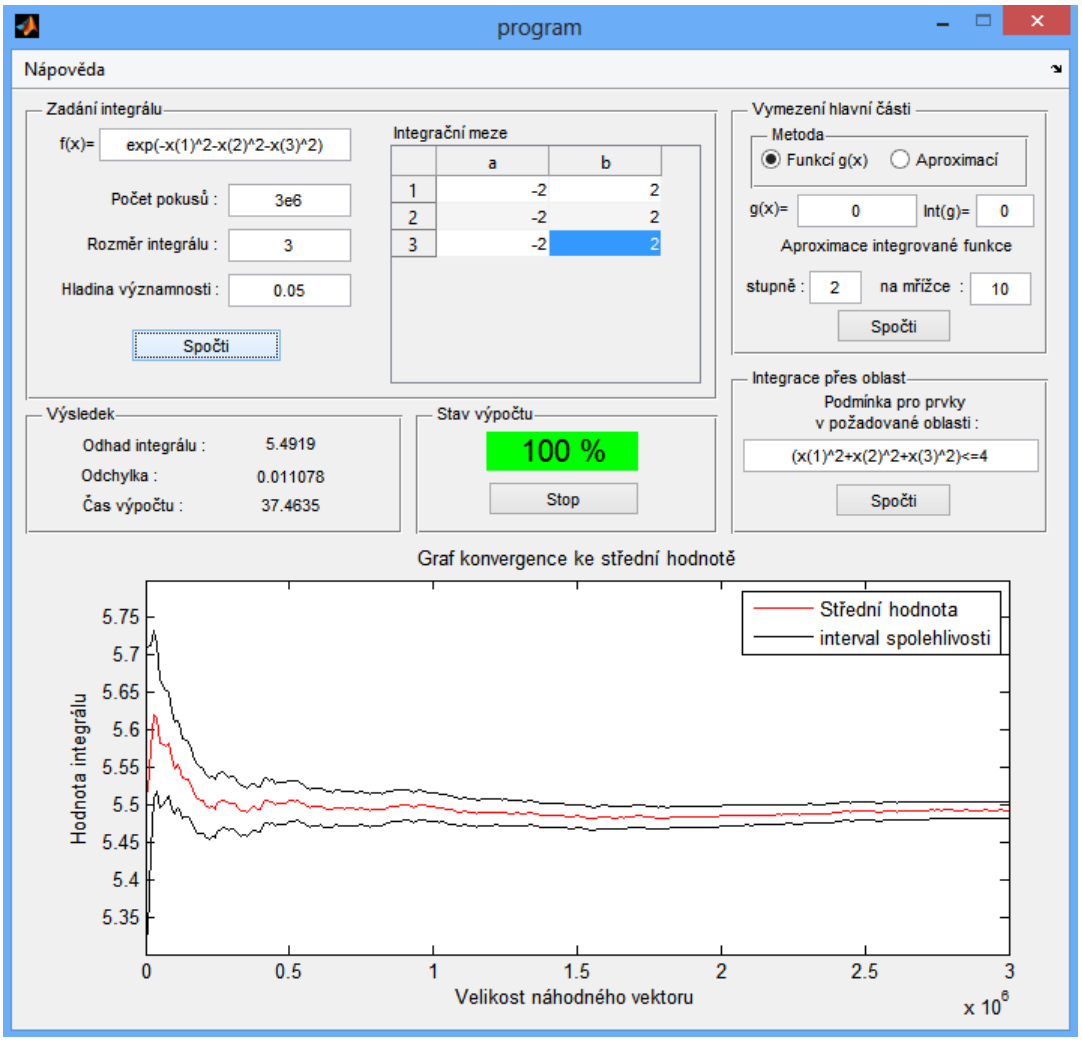
Z výsledků testování vyplývá, že interval spolehlivosti skutečně klesá, avšak výpočetní čas pro vyhodnocování integrované funkce $g(x) - f(x)$ s rostoucím stupněm polynomu narůstá. To má za následek přibližně stejnou účinnost metody při i bez využití vymezení hlavní části za použití aproximace funkce. △

6.4 Program

Součástí této práce je také program pro uživatelsky příjemné využití zde naprogramovaných funkcí. Program je koncipován pro řešení vícerozměrných integrálů pomocí MMC a zvládá následující úlohy:

- řešení vícerozměrných integrálů na daných mezích
- řešení vícerozměrných integrálů na daných mezích s využitím vymezení hlavní částí
  - za pomocí uživatelem zadané funkce a jejího integrálu
  - za pomoci spočtené aproximace funkce
- řešení vícerozměrných integrálů na obecné oblasti Ω

Výhodou je možnost zastavení výpočtu a zobrazení doposud získaných dat nebo zobrazení postupné konvergence k hledané hodnotě pomocí grafu výběrového průměru a spočtené odchylky v závislosti na velikosti náhodného vektoru. Více o použití je uvedeno v nápovědě programu, která se otevírá z kontextového menu. Ukázku prostředí programu můžeme vidět na obrázku 6.7.



Obrázek 6.7: Uživatelské prostředí programu

7 Řešení inženýrských úloh pomocí MMC

V případě řešení složitých inženýrských úloh se často setkáváme s neschopností řešit je klasickými analytickými metodami. V těchto případech se nám nabízí MMC, kterou jsme schopni daný problém s jistou přesností vyřešit. Zpravidla se jedná o komplexní úlohy, kde řešení analytickou cestou s užitím dnešních počítačů může trvat i stovky let.

V této části si uvedeme dvě takovéto úlohy a jejich řešení MMC.

7.1 Průchod neutronu obalem reaktoru

Řešení průchodu částic překážkami je jedna ze základních fyzikálních problematik řešených MMC. V této úloze budeme řešit průchodnost neutronů obalem reaktoru ve tvaru dutého válce. Nejprve si však zavedeme základní teorii týkající se této problematiky. Tato část čerpá z [7, 10, 8, 9].

7.1.1 Základní předpoklady

Zde si uvedeme základní předpoklady a zjednodušení úlohy, kterými se bude řídit náš model.

- Neutrony vyzářené ze zdroje záření jsou mono-energetické, tedy mají stejnou počáteční energii.
- Předpokládáme izotropní prostředí v plášti (ve všech směrech stejné fyzikální vlastnosti).
- Zanedbáváme možnost štěpení částic v plášti. Pro námi zvolené materiály (např. ^{56}Fe) je pravděpodobnost rozštěpení zanedbatelná.
- Při srážce neutronu jsou tedy dvě možnosti: neutron je absorbován, nebo se odrazí. Pravděpodobnost pohlcení je konstantní, dále si uvedeme její výpočet.
- Při odrazu neutron ztratí část své energie. Část energie ztracené je přímo úměrná energii původní.
- Plášť okolo zářiče považujeme za nekonečně dlouhý.
- Neutrony jsou vyzařovány z jednoho bodu. Jiný bod vyzařování (avšak také uprostřed pláště) má vliv pouze na místo opuštění neutronu, a nás zajímá pouze počet uniklých neutronů.
- Neutrony jsou vyzařovány náhodně do všech směrů kolmo k zářiči se stejnou pravděpodobností.
- Po vyzáření neutron beze ztráty energie a změny směru dorazí k vnitřnímu okraji pláště.
- Neutron se v reaktoru samovolně nerozpadne. Střední hodnota života neutronu je 11 minut, čas strávený v reaktoru je řádově mnohem menší, proto zanedbáváme samovolný rozpad.

Toto jsou předpoklady a vlastnosti pro které budeme dále vytvářet model.

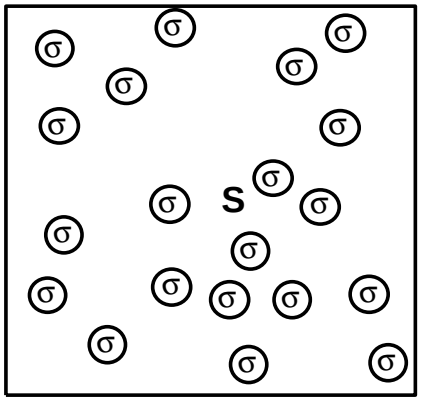
7.1.2 Chování neutronu uvnitř materiálu

Nadefinovali jsme si základní vlastnosti modelu, nyní si popíšeme průběh života neutronu uvnitř reaktoru. K tomuto si nejdříve nadefinujeme některé důležité vlastnosti materiálu ovlivňující průchod neutronu.

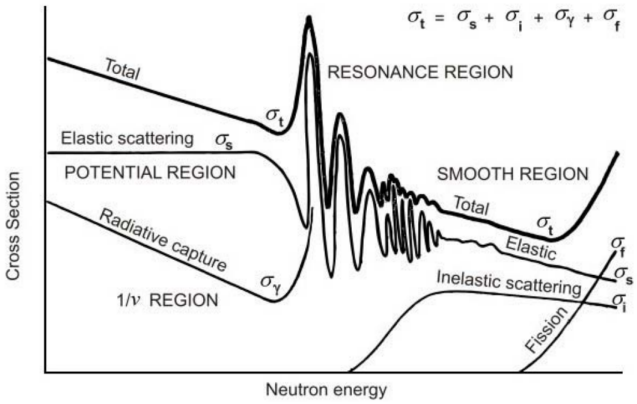
Mikroskopický účinný průřez Mějme rovinu, na níž jsou rozmístěny atomy dané látky s konstantní plošnou hustotou ρ. Pak geometrická pravděpodobnost zásahu atomu na dané ploše bude: $P = \sigma\rho$, kde σ je mikroskopický účinný průřez. Jelikož atomy jsou koule o poloměru a, můžeme makroskopický průřez vyjádřit jako obsah kruhu (řez atomu):

$$\sigma = \pi a^2.$$

Toto lze vidět na obrázku 7.1a. Z hlediska rozměrů v atomovém měřítku udáváme jednotku mikroskopického průřezu jako *barn* [bn] (1bn $= 10^{-28}\mathrm{m}^2$). Tento vztah platí pouze tehdy, když neutrony a atomy dané látky na sebe nepůsobí přitažlivými, nebo odpudivými silami. Velikosti mikroskopických průřezů se také mění v závislosti na energii neutronu, jak lze vidět na obrázku 7.1b, toto však zanedbáváme a počítáme s mikroskopickými účinnými průřezy jako s konstantami pro danou látku.



(a) Rozložení atomů na ploše



(b) Závislost mikroskopických účinných průřezů na energii neutronu [10, str. 16]

Obrázek 7.1: Ukázka mikroskopického účinného průřezu a jeho chování v závislosti na energii neutronu

Makroskopický účinný průřez Oproti mikroskopickému průřezu již charakterizuje vlastnosti látky (ne pouze atomu), kterou neutron prochází. Nejčastěji se udává objemový makroskopický průřez, vztažený na jednotku objemu. Objemový makroskopický účinný průřez spočteme jako:

$$U = \sigma\rho_V,$$

kde ρ_V je hustota částic v objemu (spočítá se z hustoty a molární hmotnosti).

Délka volného doletu Označíme λ, určuje délku trajektorie letícího neutronu do srážky s atomem prostředí. V rámci simulace života neutronu budeme potřebovat generovat její hodnoty. Pro určení jejího rozdělení nejdříve nalezneme funkci udávající počet letících neutronů v závislosti na vzdálenosti. Pro tuto funkci platí, že množství letících neutronů ve vzdálenosti x je původní počet bez počtu zachycených do této vzdálenosti:

$$\begin{aligned} f(x) &= f(0) - U\int_0^x f(z)\,\mathrm{d}z \\ f'(x) &= -Uf(x), \end{aligned} \tag{7.1}$$

řešením rovnice 7.1 například metodou variace konstanty dostaneme:

$$f(x) = C\,\mathrm{e}^{-Ux}.$$

Vidíme, že se jedná o exponenciální rozdělení. Dále spočteme koeficient C, aby funkce splňovala podmínky hustoty pravděpodobnosti:

$$\begin{aligned}
\int_0^\infty C\,\mathrm{e}^{-Ux}\,\mathrm{d}x &= 1 \\
-C\left[\frac{1}{U}\,\mathrm{e}^{-Ux}\right]_0^\infty &= 1 \\
-C\left(0-\frac{1}{U}\right) &= 1 \\
C &= U,
\end{aligned}$$

tedy $f(x) = U\,\mathrm{e}^{-Ux}$. [7]

Tedy víme, že délka volného doletu je z exponenciálního rozdělení: $\lambda \sim Exp(U)$, které již umíme simulovat.

Pohlcení neutronu při srážce Při srážce neutronu s atomem prostředí je jistá pravděpodobnost, že bude neutron pohlcen. V tomto modelu považujeme tuto pravděpodobnost za konstantní a označíme ji: p_p(pravděpodobnost pohlcení). Tato pravděpodobnost se spočítá daným poměrem makroskopických účinných průřezů, jež se dají spočíst pomocí příslušných mikroskopických účinných průřezů σ_p, σ_o uvedených ve fyzikálních tabulkách. Platí, že $U = U_p + U_o$, kde U_p je makroskopický účinný průřez pro pohlcení a U_o je makroskopický účinný průřez pro odraz. Pravděpodobnost pohlcení se tedy spočítá jako:

$$p_p = \frac{U_p}{U}.$$

Ztráta energie při srážce Pokud není neutron při srážce pohlcen, odrazí se a přitom ztratí jistou část své energie. Toto množství energie není přesně určeno avšak pro jeho střední hodnotu platí:

$$\Xi = -\ln\left(\frac{E}{E_0}\right); \Xi = 1 + \frac{(A_r-1)^2}{2A_r}\ln\left(\frac{A_r-1}{A_r+1}\right),$$

kde E je energie po srážce, E_0 je energie před srážkou, A_r je relativní atomová hmotnost daného materiálu. [10]

Tedy energii po srážce můžeme spočítat jako:

$$E = E_0\,\mathrm{e}^{-\Xi},$$

kde $\mathrm{e}^{-\Xi}$ je pro daný materiál konstanta a označíme ji e_{ztrata}.

Směr letu neutronu po srážce Díky požadavkům na izomorfní prostředí, je směr letu neutronu po srážce s atomem prostředí náhodný a rovnoměrně rozdělený do celého prostoru (každý prostorový směr má stejnou pravděpodobnost). Toto lze snadno simulovat využitím poznatků z příkladu 6.4, kde jsme generovali uniformní body z objemu koule. Nyní nám stačí zanedbat poloměr a získáme požadované rozdělení.

7.1.3 Řešení pomocí MMC

Nyní můžeme přistoupit k samotnému principu simulace.

Postup simulace života neutronu lze vyjádřit v následujících bodech:

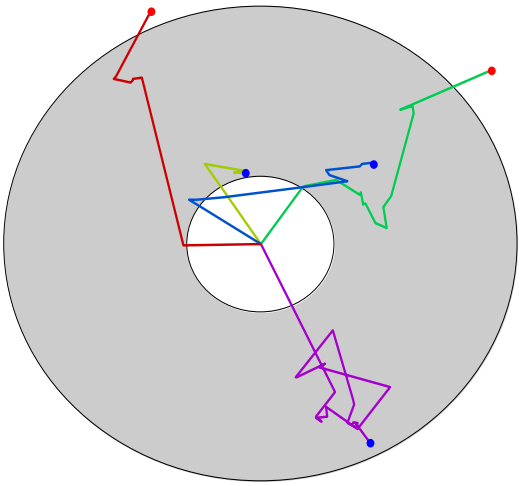
1. Vygenerujeme neutron ve středu reaktoru, přiřadíme mu počáteční energii a směr (kolmý na osu reaktoru).
2. Spočteme bod ve vnitřní hranici pláště, kam neutron dorazí a přesuneme jej.
3. Spočteme délku volného doletu a přesuneme neutron ve směru jeho letu po dané dráze.
4. Nasimulujeme srážku, zda-li je neutron pohlcen, nebo odražen. Spočítáme ztrátu energie a nasimulujeme nový směr.
5. Krok 3. a 4. opakujeme, dokud neutron není pohlcen, nebo dokud nevyletí z obalu. Pokud se odrazí a dostane zpět do dutiny obalu, necháme jej volně doletět k dalšímu vnitřnímu okraji obalu.
6. Výsledkem náhodné veličiny je 1 nebo 0, podle toho jestli neutron opustí obal.

Tedy námi modelované veličiny budou:

$$X = \begin{cases} 1, & \text{neutron unikne} \\ 0, & \text{neutron zanikne} \end{cases}, \text{ nebo } X = \begin{cases} e_{unik}, & \text{neutron unikne} \\ 0, & \text{neutron zanikne} \end{cases},$$

v případě, že zjišťujeme počet uniklých neutronů, nebo v případě, že zkoumáme uniklou energii záření.

Ukázku simulace můžeme vidět na obrázku 7.2. Zdrojový kód tohoto postupu viz výpis D.1.



Obrázek 7.2: Ukázka letu neutronů (průmět do roviny)

Jelikož je simulace života každého neutronu časově dosti náročná, je možné výpočet zefektivnit. To můžeme udělat tak, že místo jednotlivých neutronů vysíláme celý paprsek o dané hmotnosti, který při nárazech postupně ztrácí svou hmotu. Tedy místo počítání pravděpodobnosti pohlcení počítáme úbytek hmotnosti : $m = m_0(1 - p_p)$. Algoritmus pro simulaci celého svazku lze vidět ve výpise D.2.

Pro simulaci potřebujeme nejdříve spočítat jednotlivé veličiny U, p_p, e_{ztrata}, ty spočteme z veličin, které nalezneme v tabulkách (σ_p - mikroskopický účinný průřez pohlcení, σ_o - mikroskopický účinný průřez odrazu, ρ - hustota, N_A - Avogadrova konstanta, M_m - molární hmotnost, A_r - relativní atomová hmotnost).

	$\sigma_o\,[\mathsf{bn}]$	$\sigma_p\,[\mathsf{bn}]$	$\rho\,[\mathsf{kg}\cdot\mathsf{m}^{-3}]$	$N_A\,[\mathsf{mol}^{-1}]$	$M_m\,[\mathsf{kg}\cdot\mathsf{mol}^{-1}]$	A_r
^{56}Fe	0.04	1.37	7860	$6.022\cdot10^{23}$	$5.5845\cdot10^{-2}$	55.845
^{207}Pb	0.00	2.50	11340	$6.022\cdot10^{23}$	0.2072	207.2
^{58}Ni	0.23	1.27	8900	$6.022\cdot10^{23}$	$5.8693\cdot10^{-2}$	58,693

Tabulka 7.1: Fyzikální veličiny testovaných látek

Pro srovnání simulace pomocí jednotlivých neutronů a celého svazku zvolíme následující parametry: obal reaktoru s vnitřním poloměrem $a = 0.5\mathsf{m}$ a vnějším poloměrem $b = 1\mathsf{m}$ ze železa ^{56}Fe. Jednotlivé fyzikální veličiny[1] potřebné k výpočtu najdeme v tabulce 7.1.

Potřebné veličiny spočteme jako:

$$U = (\sigma_o + \sigma_p)\frac{\rho N_A}{M_m};\; p_p = \frac{\sigma_p}{\sigma_p + \sigma_o};\; e_{ztrata} = \mathrm{e}^{-1-\frac{(A_r-1)^2}{2A_r}\ln\left(\frac{A_r-1}{A_r+1}\right)} \tag{7.2}$$

a pro ^{56}Fe vychází: $U = 11.9511\mathsf{m}^{-1}, p_p = 0.0284, e_{ztrata} = 0.9652$. Nyní lze provést simulaci, bude nás zajímat pravděpodobnost úniku neutronu z reaktoru. Výsledky simulace pro 100000 pokusů lze vidět pro oba algoritmy v tabulce 7.2. Lze pozorovat, že z takto chráněného reaktoru by uniklo $39.04 \pm 0.15\%$ neutronů. Také je patrné, že metoda využívající paprsek neutronů je přibližně $2\times$ efektivnější.

	$\bar{X}$	ε	$t\,[s]$	FOM
Samostatné neutrony	0.3922	0.0030	331.18	194.83
Paprsek neutronů	0.3904	0.0015	648.3970	412.29

Tabulka 7.2: Srovnání metody simulace pro samostatné neutrony a paprsek neutronů

Dále si uvedeme příklad, kde budeme modelovat tloušťku obalu reaktoru tak, aby unikla pouze určitá energie záření.

Příklad 7.1
Zjistěte, pro jaký vnější poloměr obalu reaktoru z niklu o vnitřním poloměru $a = 0.5\mathsf{m}$, unikne pouze 1% energie z původního neutronového záření o energii $10\mathsf{MeV}$.

Nejprve spočteme potřebné veličiny podle vzorců 7.2 z fyzikálních vlastností niklu z tabulky 7.1:

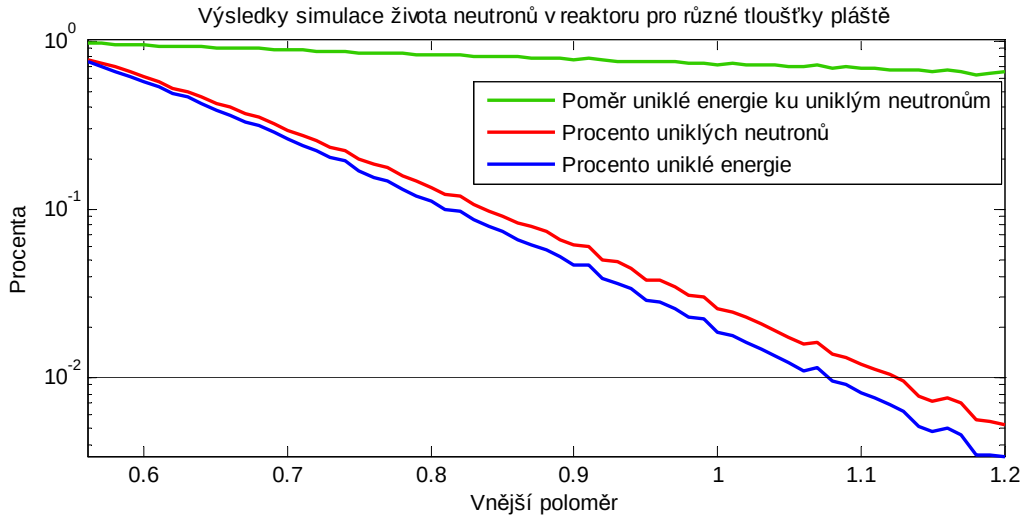
$$U = 13.6976\mathsf{m}^{-1}, p_p = 0.1533, e_{ztrata} = 0.9669.$$

Dále provedeme simulaci života neutronu v reaktoru s různými tloušťkami pláště. Volíme počáteční vnější poloměr $b = 0.56\mathsf{m}$ a pokračujeme s krokem $1\mathsf{cm}$ dokud požadovaná tloušťka pláště nebude stačit. Výsledky simulací pro 10000 pokusů lze vidět na obrázku 7.3 a vybrané hodnoty v tabulce 7.3.

[1] Hodnoty mikroskopický účinných průřezů jsou převzaty z [7].

b	0.6	0.7	0.8	0.9	1	1.1	1.2
$\frac{e_{uniklá}}{e_{vyzařená}}$	0.5758	0.2594	0.1103	0.0470	0.0186	0.0081	0.0034
$e_{uniklá}$	$5.76 \cdot 10^6$	$2.6 \cdot 10^6$	$1.1 \cdot 10^6$	$4.7 \cdot 10^5$	$1.86 \cdot 10^5$	$8.12 \cdot 10^4$	$3.42 \cdot 10^4$
$\varepsilon_{e_{uniklá}}$	$6.6 \cdot 10^4$	$5.9 \cdot 10^4$	$3.9 \cdot 10^4$	$2.5 \cdot 10^4$	$1.4 \cdot 10^4$	$8.6 \cdot 10^3$	$5.3 \cdot 10^3$

Tabulka 7.3: Výsledky simulace života neutronu pro vybrané vnější poloměry



Obrázek 7.3: Výsledky simulace života neutronů

Z obrázku 7.3 vidíme, že okolo $b = 1.1\text{m}$ hodnota procentuální uniklé energie klesá pod 1%. Přesněji první hodnota tloušťky, pro kterou je s 97,5% pravděpodobností procento uniklé energie menší než 1% je $b = 1.09\text{m}$ pro kterou z původní energie záření uniklo $8.1153 \cdot 10^4 \pm 8.7577 \cdot 10^3\text{eV}$. Tedy s 97,5% pravděpodobností unikne maximálně 0.899% vyzářené energie.

△

Průchod neutronu deskou Jedná se o mírnou modifikaci předchozí úlohy. V této úloze září zářič kolmo k desce dané tloušťky a. Algoritmus pro simulaci této úlohy získáme snadno modifikací předchozích algoritmů, lze jej vidět ve výpisu D.3. Ukázku algoritmu si ukážeme v následujícím příkladě.

Příklad 7.2
Zjistěte, jaká část energie záření, s vyzařovanou energií 10MeV, projde skrz olověnou desku o tloušťce $(a = 0.5\text{m}, a = 1\text{m})$.

Fyzikální vlastnosti olova získáme z tabulky 7.1. Spočítáme potřebné veličiny:

$$U = 8.2398\text{m}^{-1}, p_p = 0, e_{ztrata} = 0.9904.$$

Při simulaci s 100000 pokusy jsme získali odhad pravděpodobnosti průletu neutronu $\bar{p}_{prulet} = 0.3008 \pm 0.0028$ a procento proniklé energie záření $\bar{E}_{unik} = 26.79 \pm 0.25$. Provedeme simulaci ještě jednou, tentokrát pro $a = 1\text{m}$. Výsledky jsou následující: odhad pravděpodobnosti průletu neutronu $\bar{p}_{prulet} = 0.1732 \pm$

0.0023, procento proniklé energie záření $\bar{E}_{unik} = 11.81 \pm 0.17$. Jak vidíme při zvětšení tloušťky desky na dvojnásobek, pronikne pronikne stále 17% neutronů, avšak uniklá energie je již pouze 11% energie vyzářené.

△

7.2 Určení pohotovosti systému

Další klasickou problematikou řešenou MMC je výpočet pohotovosti systému. Tato část čerpá z [1].

7.2.1 Základní pojmy

Systémy Zde si uvedeme základní terminologii systémů.

- **komponenta** – základní jednotka systému, nabývá konečný počet stavů (b) (v našem případě pouze 2, v provozu/mimo provoz $(b = 1/b = 0)$) a je znám mechanizmus přechodu mezi jejími stavy
- **systém** – je soubor n komponent, které se vždy nacházejí v nějakém ze svých stavů
- **stavový vektor** – je to vektor stavů všech komponent v daném okamžiku $\vec{B} = (b_1, b_2, ..., b_n)$
- **stavový prostor** – je to množina všech možných stavových vektorů daného systému, značí se V, v našem případě $(b = 1/b = 0)$ má velikost 2^n
- **systémová funkce** – je to funkce $S\left(\vec{B}\right)$, která každému stavovému vektoru $\vec{B} \in V$ přiřazuje právě jedno reálné číslo (v našem případě pouze $1/0$)

Základy teorie spolehlivosti Jelikož se zabýváme systémy, jež mění svůj stav v čase, potřebujeme tyto změny definovat. Definice se budou týkat jedné komponenty se stavy (0/1).

- **doba do poruchy** – je náhodná veličina, která udává čas do poruchy (přechod $1 \to 0$), tedy její distribuční funkce $F_1(t)$ určuje pravděpodobnost poruchy v intervalu $(0,t)$
- **střední doba do poruchy** – zkráceně MTTF (anglicky: *mean time to failure*), udává střední hodnotu času do poruchy
- **intenzita poruch** – $\lambda_1(t)$ je hazardní funkcí náhodné veličiny *doba do poruchy*
- **doba do opravy** – je náhodná veličina, která udává čas do opravy vzniklé poruchy (přechod $0 \to 1$), tedy její distribuční funkce $F_0(t)$ určuje pravděpodobnost opravy vzniklé poruchy v intervalu $(0,t)$
- **střední doba do opravy** – zkráceně MTTR (anglicky: *mean time to repair*), udává střední hodnotu času do opravy poruchy
- **intenzita oprav** – $\lambda_0(t)$ je hazardní funkce náhodné veličiny *doba do opravy*
- **pohotovost** – je pravděpodobnost, že je v daném čase t systém v provozu, značíme $A(t)$

Pro naše potřeby použijeme dobu do poruchy a dobu do opravy pouze s exponenciálním rozdělením a tedy jednotlivé intenzity oprav a poruch jsou konstantní:

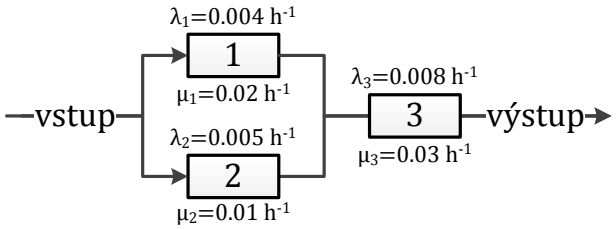
$$\lambda_1(t) = \lambda; \lambda_0(t) = \mu.$$

Potom můžeme pohotovost vyjádřit jako:

$$A(t) = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t}. \tag{7.3}$$

Princip výpočtu pohotovosti systému si ukážeme na následujícím příkladě.

Příklad 7.3
Spočtěte pohotovost systému znázorněného na obrázku 7.4 v čase $t = 100$h.



Obrázek 7.4: Schéma systému

Nejprve spočteme pravděpodobnost funkčnosti jednotlivých komponent. Dosazením do vzorce 7.3 získáme následující pravděpodobnosti:

$$p_1 = 0.84846; p_2 = 0.74104; p_3 = 0.79418.$$

Dále je třeba vyjádřit systémovou funkci daného systému, která se dá jednoduše vyjádřit jako:

$$S(b_1,b_2,b_3) = \begin{cases} 1, & \vec{B} \in \{(1,1,1),(1,0,1),(0,1,1)\} \\ 0, & \vec{B} \in \{(1,1,0),(1,0,0),(0,1,0),(0,0,1),(0,0,0)\} \end{cases} = (b_1 \cdot b_3) + (b_1 \cdot b_2),$$

kde b_1, b_2, b_3 považujeme za logické proměnné (tedy nabývají hodnot 0/1), operátor $\cdot$ považujeme za logický *and* a operátor $+$ za logický *or*. Potom pro zjištění pohotovosti celého systému musíme projít všechny stavové vektory, spočítat jejich pravděpodobnost a sečíst pravděpodobnosti vektorů, pro které $S\left(\vec{B}\right) = 1$. Řešení lze vidět v tabulce 7.4.

$\vec{B}$	$(0,0,0)$	$(0,0,1)$	$(0,1,0)$	$(0,1,1)$	$(1,0,0)$	$(1,0,1)$	$(1,1,0)$	$(1,1,1)$
$S\left(\vec{B}\right)$	0	0	0	1	0	1	0	1
$P\left(\vec{B}\right)$	0.0081	0.0312	0.0231	0.0892	0.0452	0.1745	0.1294	0.4993
$A(100)$	0.7630							

Tabulka 7.4: Počítání pohotovosti systému v čase $t = 100$

Další možností, jak analyticky spočítat pohotovost systému, je vyjádření přímo z pravděpodobnosti každé komponenty. Tedy rozdělíme systém do jednotlivých bloků, které jsou sériově či paralelně zapojeny a spočítáme pravděpodobnost funkčnosti každého bloku. V tomto případě je řešení jednoduché:

- pravděpodobnost, že bude blok komponent $1,2$ v provozu je $(1-(1-p_1)\cdot(1-p_2))$ (pro paralelně spojené komponenty je tato pravděpodobnost obecně $1-$ pravděpodobnost, že není ani jedna v provozu)
- celková pravděpodobnost tedy bude $A(t) = (1-(1-p_1)\cdot(1-p_2))\cdot p_3$

Avšak pro složitější systémy je tento postup velice náročný, proto se při rozsáhlejších systémech (v zejména těch, které nemají sériově-paralelní zapojení) používá předchozí postup.

△

Jak je vidět v příkladě 7.3 počet stavových vektorů v námi řešených systémech geometricky narůstá. Toto má za následek neschopnost řešit rozsáhlé systémy analytickými metodami (systémy o více než 40 komponentách už nejsou řešitelné). Proto využijeme MMC k řešení těchto úloh.

7.2.2 Řešení MMC

Řešení MMC je principiálně jednoduché a od analytického se liší pouze v samotném vyhodnocování systémové funkce, kde stavové vektory vybíráme náhodně. Tedy námi modelovaná náhodná veličina má tvar:

$$X = \begin{cases} 1, & \left\{\vec{B} \in V : S\left(\vec{B}\right) = 1\right\} \\ 0, & \left\{\vec{B} \in V : S\left(\vec{B}\right) = 0\right\} \end{cases}.$$

Jednotlivé vektory $\vec{B}$ generujeme podle pravděpodobností $p_1, p_2, ..., p_n$ jako vektor náhodných pokusů z Bernoulliho rozdělení. Tedy celkový postup odhadu pohotovosti systému v čase t je:

1. spočtení pravděpodobností jednotlivých komponent dle vzorce 7.3
2. pro každou pravděpodobnost p_i vygenerovat náhodný pokus z $Ber(p_i)$, čímž vznikne náhodný stavový vektor $\vec{B}_i$
3. pro náhodný stavový vektor $\vec{B}_i$ vyhodnotit systémovou funkci, čímž dostaneme jednotlivé náhodné pokusy X_i
4. opakovat od bodu 2. pro požadovanou velikost náhodného výběru
5. vyhodnotit náhodný výběr (výběrový průměr, odchylka...)

Pro zadávání systému do počítače budeme používat matici sousednosti (pro zachycení struktury systému) a tabulku hodnot MTTF a MTTR.

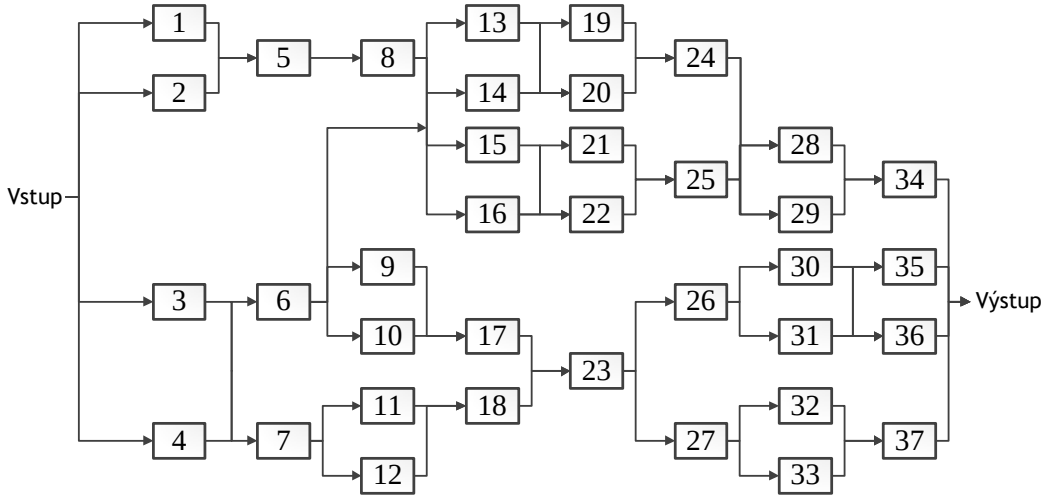
Matice sousednosti Je jedna z možností zachycení struktury systému, udává vzájemné propojení komponent. Jedná se o čtvercovou matici obsahující pouze jedničky a nuly. Každý prvek matice znamená propojení z komponenty s indexem řádku do komponenty s indexem sloupce (1 - spojení existuje, 0 - spojení neexistuje). První řádek/sloupec patří virtuální komponentě vstup a poslední řádek/sloupec patří virtuální komponentě výstup. Budeme ji používat pro řešení následujících úloh, neboť je z ní pro počítač snadněji čitelná struktura systému.

Jelikož ve většině případů dostáváme zadánu pouze strukturu systému a parametry jednotlivých komponent (MTTF a MTTR), musíme systémovou funkci vytvořit sami.

Pro tyto případy byl vytvořen algoritmus, který vytváří z matice sousednosti systémovou funkci. Toto probíhá jako postupné procházení systému od vstupu a postupné sestavování stromové struktury. Pro výslednou stromovou strukturu se následně vyberou listy, které prezentují komponentu výstup, a vytvoří se cesta ke kořenu stromu (komponenta vstup). Komponenty na této cestě stromem tvoří dráhu v struktuře zadaného systému (množinu komponent, pro které je systém v provozu). Ze všech těchto drah se na závěr složí systémová funkce jako logické součty jednotlivých drah (logický součin mezi všemi komponentami z dané dráhy). Tento algoritmus lze vidět ve výpisu E.1. Algoritmus má jistá omezení: nedokáže určit systémovou funkci systému, jež obsahuje cykly, a pro rozsáhlé systémy (počet komponent > 50) může výpočet trvat dlouho.

Jinou možností, jak počítat pohotovost systému bez znalostí systémové funkce, je řešit pro každý stavový vektor funkčnost systému zvlášť. Toto lze udělat obdobně jako hledání systémové funkce, avšak není třeba zapisovat historii průchodu, což značně urychlí výpočet. Tento postup lze vidět ve výpisu E.2. Obě tyto metody si vyzkoušíme na následujícím příkladě.

Příklad 7.4
Modelujte pohotovost systému zadaného schématem na obrázku 7.5 a parametry jednotlivých komponent v tabulce 7.5 v intervalu $t \in (0, 2000)$.



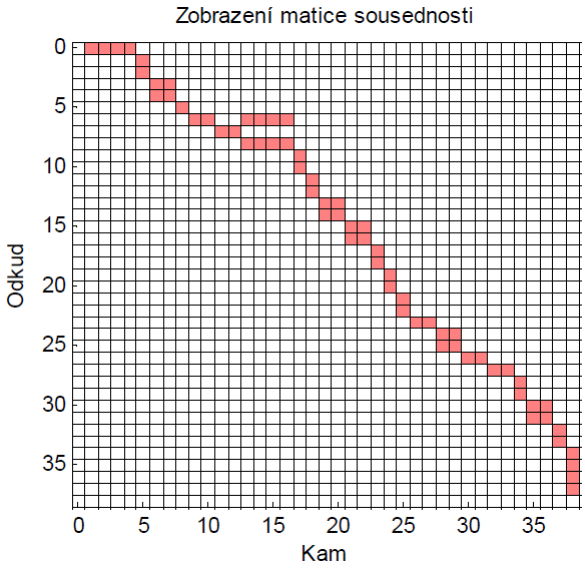
Obrázek 7.5: Zadání struktury systému

Nejdříve převedeme strukturu systému na matici sousednosti, viz obrázek 7.6.

Komponenta	MTTF	MTTR
1,2,3,4	2400	50
5,6,7	850	80
8	1300	110
9,10,11,12,13,14,15,16	400	120
17,18,19,20,21,22	950	90

Komponenta	MTTF	MTTR
23	1400	250
24,25,26,27	750	160
28,29,30,31,32,33	900	100
34,35,36,37	600	20

Tabulka 7.5: MTTF a MTTR jednotlivých komponent



Obrázek 7.6: Zobrazení matice sousednosti

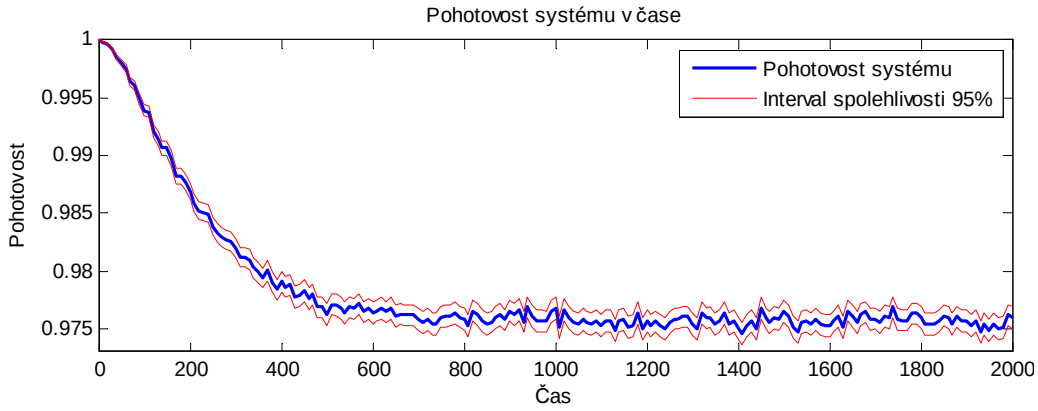
Dále vyzkoušíme, který z výše uvedených postupů bude efektivnější (jelikož daný systém nemá ve své struktuře cykly, můžeme použít oba). Porovnání účinnosti pro 100000 pokusů lze vidět v tabulce 7.6.

metoda	$\bar{A}(1000)$	ε	t [s]	FOM
pomocí řešení průchodnosti	0.9755	$9.5876 \cdot 10^4$	14.5425	$0.27345 \cdot 10^6$
pomocí systémové funkce	0.9755	$9.5876 \cdot 10^4$	3.9651	$1.0113 \cdot 10^6$

úloha	t [s]
spočtení systémové funkce	0.072762

Tabulka 7.6: Porovnání metod řešení pohotovosti systému

Jelikož je systémová funkce sestavena velmi rychle a její vyhodnocování je efektivnější než řešení průchodnosti systému, použijeme pro modelování pohotovosti právě tento postup. Pro modelování pohotovosti systému zvolíme diskrétní krok 10 a velikost náhodného výběru pro každou hodnotu času 100000, tedy spočteme pohotovost v časech $t \in (0, 10, 20, .., 2000)$. Výsledky simulace včetně 95% intervalu spolehlivosti lze vidět na obrázku 7.7 pohotovost systému ve vybraných časech v tabulce 7.7.



Obrázek 7.7: Pohotovost systému v čase $t \in (0, 2000)$

t	0	100	200	300	400	500	600	700	800
$\bar{A}(t)$	1	0.9939	0.9869	0.9820	0.9791	0.9762	0.9764	0.9758	0.9759
ε	0	0.0005	0.0007	0.0008	0.0009	0.0009	0.0009	0.001	0.001

Tabulka 7.7: Pohotovost systému v čase

Jak je vidět z výsledků, hodnota pohotovosti systému se po přibližně 800h ustálí. Hodnota pohotovosti systému v čase 2000h je 0.976 ± 0.0009.

△

8 Paralelní implementace metody Monte Carlo

Jelikož v simulacích prováděných MMC mnohdy potřebujeme značné množství pokusů, může být jejich realizace časově náročná. Z tohoto důvodu je potřeba rozložit daný výpočet do více částí, jež se mohou vykonávat paralelně. V případě MMC je tento problém principiálně jednoduchý, neboť jednotlivé náhodné pokusy mohou být prováděny odděleně.

V této části se dále budeme zabývat paralelní implementací vícerozměrné integrace a řešení problému průchodu neutronu materiálem. K řešení použijeme následující technologie:

- Matlab `parfor` – je součástí *„Matlab parallel toolbox“* a umožňuje automaticky rozdělit části `for` cyklu mezi jednotlivé procesory v podobě předem vytvořených *„Matlab workers“*
- OpenMP – jedná se o technologii, která umožňuje snadné paralelní programování v jazyce C++ pomocí stanovených direktiv (např: `#omp parallel for`)
- NVIDIA CUDA – (Compute Unified Device Architecture) umožňuje používat výpočetní sílu grafické karty (GPU), programuje se v jazyku založeném na C++, avšak přináší jistá omezení závislá na povaze grafické karty (omezená paměť, nutnost segmentace výpočtů atd.), budeme používat dva přístupy využití:

  1. CUDA kernel volaný z prostředí Matlabu, je nutné nejprve vytvořit soubor PTX (Parallel Thread Execution) pomocí externího kompilátoru (NVIDIA CUDA Compiler)
  2. výpočet na grafické kartě pomocí *„Matlab parallel toolbox“* zejména funkce `arrayfun`

8.1 Integrace

Pro paralelizaci integrace zvolíme technologii CUDA pomocí *„Matlab parallel toolbox“*. Důvodem je velice snadná implementace a také možnost využit *„function handles“*, které umožňují jednoduché zadání integrované funkce, tedy v prostředí Matlabu se bude paralelní funkce chovat pro uživatele stejně.

Implementaci lze vidět ve výpisu 8.1.

```
1  function [ NV ] = MC_CUDA_integrate( f, o, n )
2  dim=length(o)/2;          % dimenze integralu
3  NV_body=cell(1,dim);      % pozice pro jednotlive dim.
4  for i=1:dim               % vytvoreni nahodnych cisel v prostoru danym [o]
5      NV_body{i}= gpuArray.rand(1,n,'single')*(o(2*i)-o(2*i-1))+o(2*i-1);
6  end                       % jednotlive pole se alokoji na GPU
7  f_x=1/prod(o(2:2:end)-o(1:2:end)); % spocteni hodnoty hustoty p. unif. roz.
8  NV_g=arrayfun(f,NV_body{:});     % spocteni funkcni hodnoty na GPU
9  NV_g=NV_g/f_x;                   % deleni hustotou pravdepodobnosti
10 NV=double(gather(NV_g));         % stahnuti dat z GPU
```

Výpis 8.1: Paralelní implementace pomocí CUDA

Porovnání výpočtu s původním sériovým algoritmem pro sadu integrálů:

$$\int_{-2}^{2} \mathrm{e}^{-x^2}\,\mathrm{d}x, \int_{-2}^{2}\int_{-2}^{2} \mathrm{e}^{-x^2-y^2}\,\mathrm{d}x\,\mathrm{d}y, \int_{-2}^{2}\int_{-2}^{2}\int_{-2}^{2} \mathrm{e}^{-x^2-y^2-z^2}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \ldots,$$

lze vidět v tabulce 8.1.

	Matlab - jednovláknově			CUDA		
dimenze	$\bar{E}$	ε	čas [s]	$\bar{E}$	ε	čas [s]
2	3.1116	0.0024353	18.465	3.1128	0.0024354	0.1715
3	5.4895	0.0060637	22.744	5.491	0.0060664	0.26157
4	9.6832	0.014371	22.615	9.6995	0.014397	0.36792
5	17.085	0.033261	22.257	17.076	0.033199	0.46108
6	30.185	0.075862	26.086	30.185	0.075912	0.58254
7	53.212	0.17169	25.725	53.224	0.17154	0.52745
8	93.747	0.38792	25.357	93.81	0.38821	0.63173
9	165.55	0.87507	26.912	165.64	0.87182	0.68873
10	291.95	1.9654	29.347	292.74	1.9477	0.78515

Tabulka 8.1: Srovnání integrace pro sériový výpočet na CPU a výpočet na GPU

Jak lze vidět, výpočet pomocí technologie CUDA je přibližně 37× rychlejší než původní sériový.

8.2 Simulace neutronu v reaktoru

Pro simulaci neutronu bylo postupně implementováno více algoritmů, konkrétně:

- Matlab `parfor` s upravenou verzí původní implementace
- OpenMP implementace pomocí tříd (Neutron, vektor, atd.), poté pomocí jediné funkce
- CUDA kernel

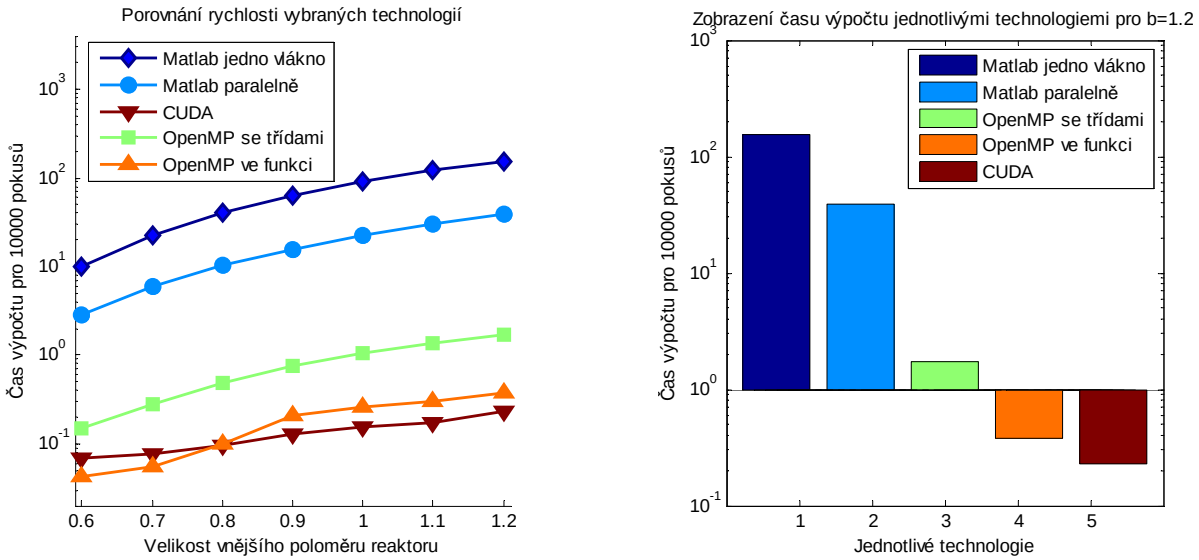
Všechny implementace kromě CUDA se významně neliší. Implementace v CUDA má několik modifikací z důvodu specifických požadavků výpočtů na GPU. Konkrétně byly provedeny následující modifikace:

- výpočet probíhá v konstantním počtu vláken po jednotlivých krocích (simulace volného letu a srážky) - neboť CUDA má specifické požadavky na dálku trvání jednotlivých vláken
- po daném počtu kroků se vždy zkontrolují simulace v jednotlivých vláknech, dokončené se uloží a na jejich místě se odstartuje nová simulace
- náhodná čísla pro výpočty délek volného letu a směru letu po odrazu se generují předem a zasílají se do CUDA kernelu jako parametr (3 pro každé vlákno)
- když počet vykonaných + vykonávaných simulací bude dostatečný, dokončí se vykonávané simulace a spočte se celkový výsledek
- jelikož CUDA pracuje především se *„single precision“* (což umožňuje pouze omezenou přesnost) je také zaveden parametr pro zanedbávání zbytkové energie svazku (pokud svazku zbude méně energie než je daná hranice, tak se jeho simulace ukončí), toto urychluje výpočet především při dlouhém životu neutronu s mnoha srážkami

Zdrojové kódy všech implementací lze nalézt na přiloženém CD.

Testy jednotlivých implementací jsou provedeny pro reaktor z niklu s charakterizujícími veličinami: $U = 13.6976\text{m}^{-1}, p_p = 0.1533, e_{ztrata} = 0.9669$ o vnitřním poloměru $a = 0.5\text{m}$, energii svazku $e = 10\text{MeV}$ a vnějších poloměrech $b = (0.6, 0.7, ..., 1.2)\,\text{m}$. Všechny výsledky lze nalézt v příloze F.

Celkovou rychlost jednotlivých implementací můžeme vidět na obrázku 8.1.



(a) Graf výpočetního času implementací

(b) Porovnání výpočetního času pro $b = 1.2$

Obrázek 8.1: Výpočetní čas jednotlivých implementací v závislosti na vnějším poloměru

Jak lze vidět z výsledků simulací, nejméně efektivní je implementace v Matlabu. Značně efektivnější je OpenMP a to v jednodušší variantě pomocí jediné funkce. Nejrychlejší je implementace pomocí CUDA: přibližně 670× rychlejší než Matlab jedno-vláknově a přibližně 8× rychlejší něž OpenMP.

9 Závěr

Hlavním cílem této práce bylo seznámení s metodou Monte Carlo a jejími aplikacemi na vybrané inženýrské úlohy. Práce řeší všechny body uvedené v zadání a lze rozdělit na teoretickou a praktickou část.

V teoretické části byly stanoveny základní poznatky z teorie pravděpodobnosti, statistiky a principy metody Monte Carlo včetně odhadu chyby řešení.

V praktické části se nejdříve věnujeme základům simulace diskrétních a spojitých rozdělení včetně ukázek a porovnání různých přístupů. Dále řešení vícerozměrných integrálů pomocí metody Monte Carlo včetně srovnání s řešením pomocí Gaussových kvadratur, kde se ukázalo, že od 6-ti rozměrných integrálů je metoda Monte Carlo vhodnějším nástrojem. V rámci části o integraci byl také ukázán princip urychlení integrace pomocí metody vymezení hlavní části a ukázka integrace přes obecné oblasti. Nakonec byly řešeny dvě inženýrské úlohy: problém průchodu neutronu materiálem, kde jsme viděli skutečnou sílu metody Monte Carlo při řešení komplexních problémů a modelování pohotovosti systémů v čase.

Součástí práce byla také implementace všech řešených úloh v jazyce Matlab. Vedle jednotlivých algoritmů byl také vytvořen program, který dovoluje ozkoušení integrace metodou Monte Carlo skrze uživatelské prostředí.

Část práce byla také věnována paralelní implementaci vícerozměrné integrace a problému průchodu neutronu materiálem. K tomuto byly použita technologie CUDA a OpenMP. Urychlení výpočtu bylo značné, například při použití technologie CUDA na problém průchodu neutronu materiálem jsme dostali 670 krát rychlejší výpočet oproti původní implementaci.

Další vývoj práce může spočívat ve zpřesňování metody Monte Carlo pomocí metod redukce rozptylu a dále v řešení integrálních rovnic pomocí metody Monte Carlo s aplikací na řešení spolehlivosti systému.

10 Reference

[1] DUBI, A. *Monte Carlo Applications in Systems Engineering.* Wiley, 2000. ISBN 0-471-98172-9.

[2] MARSEGUERRA, Marzio , ZIO, Enrico , *Basic of the Monte Carlo Method with Application to System Reliability*, LiLoLe- Verlag GmbH, 2002. ISBN 3-934447-06-6.

[3] LITSCHMANNOVÁ, Martina. *Vybrané kapitoly z pravděpodobnosti* [online]. VŠB – TU Ostrava, Fakulta elektrotechniky a informatiky, 2011. Dostupné na <mi21.vsb.cz> [citováno 10. 3. 2013]

[4] BRIŠ, R., LITSCHMANNOVÁ, M., *Statistika I,* elektronické skriptum VŠB – TU Ostrava, Fakulta elektrotechniky a informatiky, 2004

[5] LITSCHMANNOVÁ, Martina. *Úvod do statistiky* [online]. VŠB – TU Ostrava, Fakulta elektrotechniky a informatiky, 2011. Dostupné na <mi21.vsb.cz> [citováno 10. 3. 2013]

[6] KROESE, Dirk P., TAIMRE, Thomas, BOTEV, Zdravko I. *Handbook of Monte Carlo Methods.* Wiley, 2011. ISBN 978-0-470-17793-8.

[7] FABIAN, František, KLUIBER, Zdeněk. *METODA MONTE CARLO a možnosti jejího uplatnění.* 1. vyd. Praha: PROSPEKTRUM, 1998. ISBN 80-7175-058-1.

[8] ECKHARDT, R., *Stan Ulam, John von Neumann and the Monte Carlo Method*. Los Alamos Science, Special Issue (1987),

[9] HENDRICKS, J. S., *A Monte Carlo Code for Particle Transport*. Los Alamos Science (1994)

[10] CHAPLIN, R. A., *NUCLEAR INTERACTIONS, in Nuclear Energy Materials and Reactors*. Eolss Publishers, Oxford , UK, 2007. <http://www.eolss.net> [citováno 18. 4. 2013]

A Zdrojový kód inverzní transformace pro diskrétní rozdělení

```
function [ CDF ] = CDF_c( f, eps)
% tato funkce vytvori tabulku hodnot distribucni funkce
j=0;                        % iterator (hodnota pro kterou pocitame pr.)
temp(1)=f(j);               % pomocna promena urcuje zacatek nad eps
while temp(j+1)<eps
    j=j+1;
    temp(j+1)=f(j)+temp(j);
end
a=j;                        % dolni hranice hodnot
CDF(1,1)=f(a);
while CDF(1,j-a+1)<(1-eps)
    j=j+1;
    CDF(1,j-a+1)=f(j)+CDF(1,j-a);
end
CDF=[a:j;CDF];              % prirazeni hodnot k pravdepodobnostem
```

Výpis A.1: Vytvoření tabulky distribuční funkce

```
function [ NV ] = discrete_simul( CDF, n )
% simulace diskretni promene pomoci bin. vyhledavani
% CDF - tabulka distribucni funkce
% n - velikost nahodneho vyberu
NV=zeros(1,n);                  % alokace vektoru simulovanych cisel
CDF=[0 CDF(1,:); 0 CDF(2,:)];   % pridani krajniho bodu pro snadnejsi vyhled.
for i=1:n
    xi=rand();                  % nahodne cislo
    left=2;                     % dolni hranice
    right=length(CDF);          % horni hranice
    while true
        index=floor((right-left)/2+left); % stred intervalu
        if left==index                    % narazili jsme na konec intervalu
             NV(i)=CDF(1,index+1);
             break;
        end
        if CDF(2,index)>=xi
            if CDF(2,index-1)<xi
                NV(i)=CDF(1,index);       % nalezeno
                break;
            else
                right=index;              % zmena hranice
            end
        else
            left=index;                   % zmena hranice
        end
    end
end
```

Výpis A.2: Simulace diskrétní náhodné veličiny pomocí inverzní transformace

B Aproximace funkcí pomocí nejmenších čtverců

```
function [ aproc_func,vysledek,integral] = aproximace(f, o, k, n)
%tato funkce spocte aproximaci polynomemfunkce lib. dim.
% f - funkce k aproximaci
% o - oblast pro aproximaci
% k - stupen polynomu
% n - pocet deleni site pro aproximaci
dim=length(o)/2;              % dimenze funkce
bound=zeros(n,dim);           % deleni os jednotl. dim.
for i=1:dim
    bound(:,i)=linspace(o(2*i-1),o(2*i),n);
end
INDEX=ones(1,dim);            % urceni pozice v siti bodu
temp=0:n:(dim*n-n);           % pom. prom k indexaci matice
sit_bodu=zeros(n^dim,dim);    % sit bodu k aprox.
for j=1:n^dim                 % vytvoreni site bodu
    sit_bodu(j,:)=bound(INDEX+temp);
    INDEX(1)=INDEX(1)+1;
    for l=1:dim-1
        if INDEX(l)<=n
            break
        end
        INDEX(l)=1;
        INDEX(l+1)=INDEX(l+1)+1;
    end
end
base=comb_o(dim+1,k);         % bazove funkce tvori kombinace s opakováním
base=pocty(base,dim+1);       % spocitani mocniny kazde promenne
base_f=bazov_funkce(base);    % vytvoreni bazovych funkci
m=length(base_f);             % pocet bazovych funkci
hod_baz=zeros(n^dim,m);       % hodnoty baz. fun. na siti
for i=1:m
    for j=1:n^dim
        hod_baz(j,i)=base_f{i}(sit_bodu(j,:));
    end
end
hod_fun=zeros(n^dim,1);       % hodnoty aprox. funkce v siti
for j=1:n^dim
    hod_fun(j)=f(sit_bodu(j,:));
end
A=zeros(m);                   % Grammova matice
for i=1:m
    for j=i:m
        A(i,j)=sum(hod_baz(:,i).*hod_baz(:,j));
    end
end
A=A+triu(A,1)';               % vyuziti symetr. natice
b=zeros(1,m);                 % vektor pravych stran
for i=1:m
```

```
49      b(i)=sum(hod_baz(:,i).*hod_fun);
50  end
51  vysledek=A\b';                % spocteni soustavy norm. rovnic
52  aproc_func=polynom_base(base,vysledek); % vytvoreni polyn.
53  integral=int_polynom(base,vysledek,o);  % spočtení integralu polynomu
```

Výpis B.1: Aproximace vícedim. funkcí

```
1  function [ Integral ] = int_polynom( base, coeff,o)
2  % funkce urcena pro integraci vicerozmerneho polynomu
3  % base - tabulka s exponenty bazovych funkci [2 3 0;1 0 0;0 0 1]=x^3*y^2,y,1
4  % coeff - vektor ciselnych koeficientu prad danymy bazovymi funkcemi polyn.
5  % o - integracni meze : [a b c d]-> x->(a,b),y->(c,d)
6  [n,k]=size(base);    % pocet promenych a bazovych funkci
7  base=base+1;         % integrace bazovych funkci
8  coeff_int=arrayfun(@(x)1/x,base(:,1:end-1)); % spocteni koef. vznikleho int.
9  for i=1:n
10     temp=1;
11     for j=1:k-1
12         temp=temp*(o(2*j)^base(i,j)-o(2*j-1)^base(i,j)); % pocitani hodnot pro
13     end                                                  % integ. meze
14     coeff(i)=coeff(i)*prod(coeff_int(i,:));
15     coeff(i)=coeff(i)*temp;
16 end
17 Integral=sum(coeff);  % suma integralu vsech bazovych funkci
```

Výpis B.2: Mechanická integrace polynomu

C Zdrojove kody pro integraci

```
function [ NV ] = MC_integrate( f, o, n )
% funkce pro integraci vicerozmernych integralu pomoci MMC
% f - vicerozmerna funkce pro integraci pr. f=@(x)x(1)^2+sin(x(2))
% o - pole urcujici integracni oblast omega pr. omega=(a,b)x(c,d)->o=[a b c d]
% n - pozadovana velikost nah. vyberu se kterym pocitame

NV_t=U_mult(o,n);       % vytvoreni NV bodu v danem prostoru pomoci unif. r.
NV=zeros(1,n);          % alokace vektoru NV funkce nah. vel.
for i=1:n
    NV(i)=f(NV_t(:,i)); % funkce nah. vel.
end
f_x=1/prod(o(2:2:end)-o(1:2:end)); % spocteni hodnoty hustoty p. unif. roz.
NV=NV/f_x;              % deleni hust. prav. (je konstantni)
```

Výpis C.1: Vícerozměrná integrace pomocí MMC

```
function [ i ] = Gauss( f, o, k )
% tato funkce pocita integral funkce f pomoci Gaussovych kvadratur
% f - integrovana funkce, pozadovan vektorovy zapis pr. f=@(x)x(1)+x(2)...
% o - integracni meze pr. 3-dim. [0 1 0 1 0 1] (integrace pres jed. krychli
% k - stupen Gaussovy kvadratury
[z,w]=GL_zw(k);             % body a vahy Legendrovych polynomu pro dane k
n=length(o)/2;              % dimenze integralu
bound=zeros(k,n);           % predalokace site pro integraci
for i=1:n                   % vytvoreni site pro integraci transformaci
    bound(:,i)=(o(2*i-1)+o(2*i))/2 + z*(o(2*i)-o(2*i-1))/2;
end                         % z korenu Leg. polynomu
INDEX=ones(1,n);            % pole proiteraci (pozice v siti)
temp=0:k:(n*k-k);           % pom. pole pro indexaci matice jako vektoru
f_h=zeros(1,k^n);           % pole hodnot vazenych hodnot v bodech site
for j=1:k^n
    f_h(j)=f(bound(INDEX+temp));  % funkci hodnota bodu
    f_h(j)=prod(w(INDEX))*f_h(j); % prenasobeni vahou
    INDEX(1)=INDEX(1)+1;
    for l=1:n-1             % urceni pozice v siti
        if INDEX(l)<=k
            break
        end
        INDEX(l)=1;
        INDEX(l+1)=INDEX(l+1)+1;
    end
end
i=sum(f_h);
for j=1:n                   % zpetna transformace
    i=i*(o(2*j)-o(2*j-1))/2;
end
```

Výpis C.2: Vícerozměrná integrace pomocí Gaussových kvadratur

```
function [ NV ] = Int_Piece( f, a, b, k, n )
% funkce pro integraci pomocí po castech uniformniho rozdeleni
% f - funkce pro integraci
% a,b - integracni meze
% k - pocet useku po castech uniformniho rozdeleni
% n - velikost nah. vyberu
[NV_t,f_val]=Piece_dist(f,a,b,k,n); % vytvoreni NV a prisl hodnot hust. prav.
NV=arrayfun(f,NV_t);                % spocteni hodnoty funkce nah. veliciny
NV=NV./f_val;                       % deleni hustotou pravdep.

function [NV,f_val]=Piece_dist(f,a,b,k,n)
%funkce pro vytvoreni nah vyberu z po castech uniformniho rozdeleni (PCUR)
% f - funkce urcujici tvar rozdeleni
% a,b - meze intervalu pro ktery vytvarime nah vyber
% k - pocet useku po castech uniformniho rozdeleni
% n - velikost nah. vyberu
element=(b-a)/k;                            % velikost jednoho useku
bound=linspace(a+element/2,b-element/2,k);  % pole se stredy jedn. useku
f_x=arrayfun(@(x)abs(f(x)),bound);          % funkci hod. v danych bodech
disc_p=f_x/sum(f_x);                        % prepocet na d. pravdep. fun.
F_x=[(1:length(bound));disc_p];             % vytvoreni tabulky
disc_p=disc_p/element;                      % hodnoty pravdep. rozdeleni
for i=2:k
    F_x(2,i)=F_x(2,i)+F_x(2,i-1);           % vytvoreni distrib. funkce
end
NV_t=discrete_simul( F_x, n );              % vytvoreni NV dane dist. f.
NV=zeros(1,n);
f_val=zeros(1,n);
for i=1:n
    NV(i)=rand()*element-element/2+bound(NV_t(i)); % nahod. pok. PCUR
    f_val(i)=disc_p(NV_t(i));               % hodnota hust. pravd. nah. pok.
end
```

Výpis C.3: Integrace pomocí po částech uniformního rozdělení

```
function [ NV] = Int_sphere( f, R, n )
% integrace funkce f pro oblast ve tvaru koule
% f - integrovana funkce
% R - polomer koule
% n - velikost NV
f_x=1/(4/3*pi*R^3);     % spocteni hustoty pr. jako 1/objem koule
NV=zeros(1,n);
for i=1:n
    NV(i)=f(U_sphere(R))/f_x;
end

function [ x ] = U_sphere( R )
% simulace uniformniho rozdeleni v kouli o polomeru R
r=R*rand()^(1/3);               % simulace polomeru ve sfer. sour.
fi=2*pi*rand();                 % simulace uhlu fi ve sfer. sour.
```

```
16 theta=asin(2*rand()-1);          % simulace uhlu theta ve sfer. sour.
17 x=zeros(1,3);
18 x(1)=r*cos(fi)*cos(theta);       % prevedeni do Kartezkych souradnic
19 x(2)=r*sin(fi)*cos(theta);
20 x(3)=r*sin(theta);
```

Výpis C.4: Integrace přes oblast ve tvaru koule

```
1  function [ NV ] = MC_integrate_o( f, o, n, omega)
2  % funkce pro integraci vicerozmernych integralu pres oblast
3  % f - vicerozmerna funkce pro integraci pr. f=@(x)x(1)^2+sin(x(2))
4  % o - pole urcujici oblast omega1, omega je podmnozinou omega1
5  % n - pozadovana velikost nah. vyberu se kterym pocitame
6  % omega - funkce vracejici 1/0 jestlize dany bod patri/nepatri do omega
7  NV_t=U_mult(o,n);        % vytvoreni NV bodu v omega1 pomoci unif. r.
8  NV=zeros(1,n);           % alokace vektoru NV funkce nah. vel.
9  for i=1:n
10     if omega(NV_t(:,i))==1  % urceni, zdali bod patri do omega
11         NV(i)=f(NV_t(:,i)); % funkce nah. vel.
12     else
13         NV(i)=0;
14     end
15 end
16 f_x=1/prod(o(2:2:end)-o(1:2:end)); % spocteni hodnoty hustoty p. unif. roz.
17 NV=NV/f_x;                % deleni hust. prav. (je konstantni)
```

Výpis C.5: Vícerozměrná integrace pomocí MMC přes oblast Ω

D Simulace života neutronu

```
classdef Neutron < handle
    properties
        v                   % vektor smeru letu
        r                   % pozice neutronu
        pohlcen=false;      % status, zdali je pohlcen
        unikl=false;        % status, zdali unikl
        energie             % energie neutronu
        historie            % historie pozic
    end
    methods
        function obj = Neutron(ener)    % konstuktor, jako vstup energie
            obj.energie=ener;
            obj.r=[0 0 0];              % pocatecni poloha v pocatku
            xi=2*pi*rand();
            obj.v=[cos(xi) sin(xi) 0];  % pocatecni smer kolmo na osu z
            obj.historie=obj.r;         % zapsani prvni polohy
        end
        function zivot(obj)             % simulace zivota neutronu
            global a b  % zpristupneni rozmeru valce jako glob. promenych
            while ~(obj.pohlcen||obj.unikl) % simulace dokud neunikne/nezanik.
                if norm(obj.r(1:2))<a-eps   % zdali je uvnitr plaste
                    obj.let_vnitrni_okraj;
                    obj.historie=[obj.historie;obj.r];
                else
                    if norm(obj.r(1:2))>=b  % zdali neuniknul
                        obj.unikl=true;
                        return
                    end
                    obj.volny_let;          % simulace letu uvnitr plaste
                    obj.historie=[obj.historie;obj.r];
                    if norm(obj.r(1:2))<a-eps % zdali se nevratil do dutiny
                        obj.let_vnitrni_okraj;
                        obj.historie=[obj.historie;obj.r];
                    else
                        if norm(obj.r(1:2))>=b % zdali neopustil reaktor
                            obj.unikl=true;
                            return
                        end
                        obj.srazka;
                    end
                end
            end
        end
        function let_vnitrni_okraj(obj)  % dopraveni neutronu k vnitrni
            global a                     % hranici plaste
            p=obj.r(1:2);
            s=obj.v(1:2);
            t=(-dot(p,s)+sqrt(dot(p,s)^2-dot(s,s)*(dot(p,p)-a^2)))/dot(s,s);
```

```
49              obj.r=obj.r+obj.v*t;
50          end
51          function volny_let(obj)          % simulace volneho letu v plasti
52              global U
53              lambda=Exp(U);
54              obj.r=obj.r+obj.v*lambda;
55          end
56          function srazka(obj)             % simulace srazky
57              global p_pohlceni e_ztrata
58              if rand()<p_pohlceni
59                  obj.pohlcen=true;
60                  return
61              end
62              obj.energie=obj.energie*e_ztrata;  % snizeni enrgie
63              obj.v=U_3D_direction();     % novy smer neutronu
64          end
65      end
66  end
```

Výpis D.1: Simulace života neutronu v reaktoru pro jednotlivé neutrony

```
1   classdef Neutron1 < handle
2       properties
3           v                   % vektor smeru  paprsku
4           r                   % pozice paprsku
5           unikl=false;        % status, zdali unikl
6           energie             % energie neutronu
7           mass=1;             % hmota paprsku
8       end
9       methods
10          function obj = Neutron1(ener)   % konstuktor, jako vstup energie
11              obj.energie=ener;
12              obj.r=[0 0 0];              % pocatecni poloha v pocatku
13              xi=2*pi*rand();
14              obj.v=[cos(xi) sin(xi) 0];  % pocatecni smer kolmo na osu z
15          end
16          function zivot(obj)             % simulace zivota neutronu
17              global a b  % zpristupneni rozmeru valce jako glob. promenych
18              while ~(obj.unikl)          % simulace dokud neunikne/nezanik.
19                  if norm(obj.r(1:2))<a-eps      % zdali je uvnitr plaste
20                      obj.let_vnitrni_okraj;
21                  else
22                      if norm(obj.r(1:2))>=b     % zdali neuniknul
23                          obj.unikl=true;
24                          return
25                      end
26                      obj.volny_let;             % simulace letu uvnitr plaste
27                      if norm(obj.r(1:2))<a-eps  % zdali se nevratil do dutiny
28                          obj.let_vnitrni_okraj;
29                      else
```

```
                        if norm(obj.r(1:2))>=b % zdali neopustil
                            obj.unikl=true;
                            return
                        end
                        obj.srazka;
                    end
                end
            end
        end
        function let_vnitrni_okraj(obj)  % dopraveni neutronu k vnitrni
            global a                     % hranici plaste
            p=obj.r(1:2);
            s=obj.v(1:2);
            t=(-dot(p,s)+sqrt(dot(p,s)^2-dot(s,s)*(dot(p,p)-a^2)))/dot(s,s);
            obj.r=obj.r+obj.v*t;
        end
        function volny_let(obj)          % simulace volneho letu v plasti
            global U
            lambda=Exp(U);
            obj.r=obj.r+obj.v*lambda;
        end
        function srazka(obj)             % simulace srazky
            global p_pohlceni e_ztrata
            obj.mass=obj.mass*(1-p_pohlceni);  % pohlceni casti hmot. svazku
            obj.energie=obj.energie*e_ztrata;  % snizeni enrgie
            obj.v=U_3D_direction();      % novy smer neutronu
        end
    end
end
```

Výpis D.2: Simulace života neutronu v reaktoru pro svazek neutronů

```
classdef Neutron2 < handle
    properties
        v                   % vektor smeru letu
        r                   % pozice neutronu
        mass=1;             % hmotnost svazku
        unikl=false;        % status, zdali unikl
        odrazil=false;      % zdali se odrazil od desky
        energie             % energie neutronu
    end
    methods
        function obj = Neutron2(ener)   % konstuktor, jako vstup energie
            obj.energie=ener;           % energie svazku
            obj.r=[0 0 0];              % pocatecni poloha v pocatku
            obj.v=[1 0 0];              % pocatecni smer kolmo na osu z
        end
        function zivot(obj)             % simulace zivota neutronu
            global a
            while ~(obj.unikl||obj.odrazil) % simulace dokud neunikne/odrazi
```

```
19                  obj.volny_let;      % simulace volneho letu
20                  if obj.r(1)<0 -eps  % zdali se neodrazil
21                      obj.odrazil=true;
22                      return
23                  end
24                  if obj.r(1)>a  +eps % zdali neuniknul
25                      obj.unikl=true;
26                      return
27                  end
28                  obj.srazka;         % simulace srazky
29              end
30          end
31          function volny_let(obj)     % simulace volneho letu v desce
32              global U
33              lambda=Exp(U);
34              obj.r=obj.r+obj.v*lambda;
35          end
36          function srazka(obj)                    % simulace srazky
37              global p_pohlceni e_ztrata
38              obj.mass=obj.mass*(1-p_pohlceni);   % ztrata hmotnosti svazku
39              obj.energie=obj.energie*e_ztrata;   % snizeni enrgie
40              obj.v=U_3D_direction();             % novy smer neutronu
41          end
42      end
43  end
```

Výpis D.3: Simulace života neutronu v desce pro svazek neutronů

E Modelování pohotovosti systémů

```
function [ vysledek ,fun] = mat2fun( M )
% funkce k vytvoreni systemove funkce z matice sousednosti
% M - matice sousednosti (1. komponenta- vstup. posledni - vystup)
n=size(M,1);                    % pocet komponent
tree=zeros(1e4,2);              % prealokace matice reprezetujici strom
i=2;                            % index pro dalsi ulozeni listu
tree(1,1)=1;                    % prvni polozka = vstup
tree(1,2)=0;                    % nema rodice
leaves=zeros(1,n);              % predalokace vektoru listu
leav_count=0;                   % pocet aktivnich listu
leaves_final=zeros(1,1e4);      % listy, ktere skoncili ve vystupu
leaves_final_count=0;           % pocet listu, ktere repr. drahu
leav_count=leav_count+1;        % pridani vstupu, jako prvni list
leaves(leav_count)=1;           % ulozeni pozice prvniho listu
while leav_count>0              % dokud jsou aktivni listy
    leaves_temp=zeros(1,n);     % aktivni listy pro dalsi pruchod
    leaves_temp_count=0;        % jejich pocet
    for l=leaves(1:leav_count)  % pro vsechny aktivni listy
        comp=tree(l,1);         % cislo komponenty ve ktere se nachazime
        for k=find(M(comp,:))   % pro vsechny komponenty, kam muzeme jit
            tree(i,1)=k;        % vytvori novy list
            tree(i,2)=l;
            if k==n             % pokud jsme v cili list ulozi do vystupnich
                leaves_final_count=leaves_final_count+1;
                leaves_final(leaves_final_count)=i;
            else                % pokud nejsme v cili list pridame pro dalsi p.
                leaves_temp_count=leaves_temp_count+1;
                leaves_temp(leaves_temp_count)=i;
            end
            i=i+1;
        end
    end
    leaves=leaves_temp;         % prochazime skrze neve aktivni listy
    leav_count=leaves_temp_count;
end
vysledek=zeros(leaves_final_count,n); % predalokace tabulky drah
for i=1:leaves_final_count
    vysledek(i,:)=find_path(tree,leaves_final(i),n); % projde strom
end
fun=vytvor_funkci(vysledek(:,2:end-1)); % vytvori function handle
%% - funkce pro nalezeni cesty od listu stromu ke koreni (draha v systemu)
function pole=find_path(tree,leaf,n)
pole=zeros(1,n);
while tree(leaf,2)~=0       % dokud nejsme u korene
    pole(tree(leaf,1))=1;   % odkaz na rodice
    leaf=tree(leaf,2);
end
%% - funkce pro sestaveni "matlab function handle" z danych drah systemu
```

```
function [ func ] = vytvor_funkci( base)
[n,k]=size(base);
temp='';            % retezec, ze ktereho se setavi function handle
for i=1:n
    temp1='';
    for j=1:k
        if base(i,j)~=0
            temp1=[temp1 'x(' num2str(j) ')&']; % logicky soucin jedn. kompon.
        end
    end
    temp=[temp1(1:end-1) '|' temp];     % logicky soucet jednotlivych drah
end
temp=['@(x)' temp(1:end-1)];
func=str2func(temp);                    % konverze textoveho retezce na funkci
```

Výpis E.1: Vytváření systémové funkce z matice sousednosti

```
function [ pruchod ] = pruchod( M, B)
prosle=[1 B 1];           % vektor proslych hodnot
aktivni=M(1,:);           % aktivní komponenty
while ~(aktivni(end)>0 || any(aktivni)==0)% dokud je kam jit nebo nalezen pruchod
    aktivni_old=aktivni;
    aktivni=sum(M(aktivni&prosle,:),1);   % kam se lze dostat z aktivnich kompon.
    prosle=~(~prosle|aktivni_old);        % jiz prosle ignorujeme (zabraneni cyklu)
end
pruchod=any(aktivni(end));                % zdali byl dosazen vystup
```

Výpis E.2: Zjištění, zdali je systém funkční pro daný stavový vektor

F Výsledky testů paralelních implementací

	Pravděpodobnost úniku		Uniklá energie		
b	$\bar{E}$	ε	$\bar{E}$	ε	čas [s]
0.6	0.607218	0.00634276	5.69388e+006	67007.1	0.131
0.7	0.291647	0.0060579	2.55977e+006	59835.6	0.287
0.8	0.134797	0.00431513	1.11067e+006	40670.9	0.479
0.9	0.0604612	0.00280063	469137	25345.9	0.716
1	0.0273969	0.00170946	198902	14767.6	1.003
1.1	0.0123343	0.00108107	85067.4	9207.56	1.331
1.2	0.00518643	0.000585519	32649.8	4700.3	1.716

Tabulka F.1: OpenMP implementace třídami pro 10000 pokusů

	Pravděpodobnost úniku		Uniklá energie		
b	$\bar{E}$	ε	$\bar{E}$	ε	čas [s]
0.6	0.605741	0.00644034	5.68596e+006	67884	0.043
0.7	0.293866	0.00604119	2.57844e+006	59649.5	0.055
0.8	0.13313	0.00430109	1.09701e+006	40492.7	0.1
0.9	0.0596346	0.00276828	461876	25010.4	0.208
1	0.028809	0.00182019	212208	15931.6	0.262
1.1	0.0115544	0.00101067	78557.3	8483.55	0.305
1.2	0.00527166	0.000647912	33955.1	5497.4	0.378

Tabulka F.2: OpenMP implementace funkcí pro 10000 pokusů

	Pravděpodobnost úniku		Uniklá energie		
b	$\bar{E}$	ε	$\bar{E}$	ε	čas [s]
0.6	0.60644	0.0062441	5.6872e+06	65948	9.878
0.7	0.29433	0.0059171	2.5828e+06	58384	22.648
0.8	0.13241	0.0041841	1.0883e+06	39407	40.782
0.9	0.061376	0.0027704	4.7672e+05	25129	63.043
1	0.027229	0.0017121	1.9872e+05	15000	90.026
1.1	0.011549	0.0010164	79191	8590.9	120.9
1.2	0.0045967	0.00056438	29010	4658	155.62

Tabulka F.3: Matlab sériový výpočet pro 10000 pokusů

	Pravděpodobnost úniku		Uniklá energie		
b	$\bar{E}$	ε	$\bar{E}$	ε	čas [s]
0.6	0.60707	0.0062287	5.6927e+06	65821	2.8771
0.7	0.29765	0.0059533	2.6151e+06	58829	5.9147
0.8	0.13817	0.004279	1.1412e+06	40294	10.286
0.9	0.060515	0.0027122	4.6739e+05	24473	15.803
1	0.028259	0.001767	2.0772e+05	15484	22.325
1.1	0.011448	0.00098176	77483	8242.7	30.265
1.2	0.0057417	0.00071735	38518	6101.7	38.567

Tabulka F.4: Matlab paralelní výpočet pro 10000 pokusů

	Pravděpodobnost úniku		Uniklá energie		
b	$\bar{E}$	ε	$\bar{E}$	ε	čas [s]
0.6	0.61032	0.0062665	5.7309e+06	66149	0.069474
0.7	0.29203	0.0059278	2.5618e+06	58558	0.077979
0.8	0.1389	0.0043082	1.149e+06	40619	0.094865
0.9	0.059218	0.0026845	4.5751e+05	24169	0.12648
1	0.025537	0.0016173	1.8398e+05	14064	0.1575
1.1	0.010465	0.00088494	69308	7251.5	0.17203
1.2	0.005207	0.00058538	33020	4691	0.23122

Tabulka F.5: CUDA pro 10000 pokusů

	Pravděpodobnost úniku		Uniklá energie		
b	$\bar{E}$	ε	$\bar{E}$	ε	čas [s]
0.6	0.60758	0.00062383	5.6999e+06	6586.2	0.53552
0.7	0.29582	0.0005948	2.5988e+06	5878.1	0.68906
0.8	0.13511	0.00042381	1.1144e+06	3989.4	1.0722
0.9	0.060467	0.00027282	4.6834e+05	2465.4	1.5583
1	0.026752	0.00016752	1.9439e+05	1458.6	2.1625
1.1	0.011815	0.00010196	80798	861.66	2.7824
1.2	0.0051519	6.0635e-05	33056	497.4	3.4376

Tabulka F.6: CUDA pro 1000000 pokusů

	Pravděpodobnost úniku		Uniklá energie		
b	$\bar{E}$	ε	$\bar{E}$	ε	čas [s]
0.6	0.608141	0.000636637	5.70576e+006	6722.22	2.551
0.7	0.29526	0.000606983	2.59356e+006	5997.02	4.014
0.8	0.135196	0.000432587	1.11522e+006	4071.61	6.826
0.9	0.0604457	0.000278638	468349	2518.54	10.615
1	0.0267078	0.00017157	194405	1495.16	16.235
1.1	0.0117727	0.000103971	80558.2	878.287	20.953
1.2	0.00516297	6.16751e-005	33124.8	503.684	26.945

Tabulka F.7: OpenMP pro 1000000 pokusů

G Příloha na CD

Přiložené CD obsahuje:

- elektronickou kopii bakalářské práce
- veškeré zdrojové kódy vzniklé v rámci této práce
- originály vložených obrázků